

日本国特許庁
JAPAN PATENT OFFICE

67471-021

Y. YAMAMOTO

July 23, 2003.

McDermott, Will & Emery

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office

出願年月日

Date of Application:

2002年 7月24日

出願番号

Application Number:

特願2002-215394

[ST.10/C]:

[JP2002-215394]

出願人

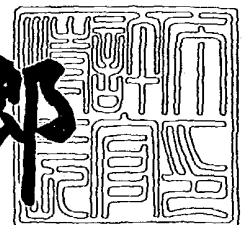
Applicant(s):

松下電器産業株式会社

2003年 3月14日

特許庁長官
Commissioner,
Japan Patent Office

太田信一郎



出証番号 出証特2003-3017282

【書類名】 特許願

【整理番号】 5037730146

【提出日】 平成14年 7月24日

【あて先】 特許庁長官 殿

【国際特許分類】 G06F 9/42

【発明者】

【住所又は居所】 大阪府門真市大字門真1006番地 松下電器産業株式会社内

【氏名】 山本 泰宜

【特許出願人】

【識別番号】 000005821

【氏名又は名称】 松下電器産業株式会社

【代理人】

【識別番号】 100090446

【弁理士】

【氏名又は名称】 中島 司朗

【手数料の表示】

【予納台帳番号】 014823

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【包括委任状番号】 9003742

【プルーフの要否】 要

【書類名】 明細書

【発明の名称】 情報処理装置、情報処理方法、およびプログラム変換装置

【特許請求の範囲】

【請求項 1】 演算に用いるデータを保持する 1 つ以上のレジスタを有し、機械語プログラムに従って処理を行うプロセッサを含む情報処理装置であって、所定の関数の呼び出しに伴いスタックメモリへ前記レジスタの内容を圧縮して退避することを示す情報が、前記機械語プログラム中に有るか否かを識別する圧縮識別手段と、

圧縮識別手段による識別結果が肯定的である場合には所定の関数の呼び出しに伴いスタックメモリへ前記レジスタの内容を圧縮して退避し、圧縮識別手段による識別結果が否定的である場合には所定の関数の呼び出しに伴いスタックメモリへ前記レジスタの内容を圧縮せずに退避する退避手段と

を備えることを特徴とする情報処理装置。

【請求項 2】 前記情報処理装置はさらに、

所定の関数の呼び出し終了に伴い前記レジスタへスタックメモリに退避されているデータを伸張して復帰することを示す情報が、前記機械語プログラム中に有るか否かを識別する伸張識別手段と、

伸張識別手段による識別結果が肯定的である場合には所定の関数の呼び出し終了に伴い前記レジスタへ前記スタックメモリに退避されているデータを伸張して復帰し、伸張識別手段による識別結果が否定的である場合には所定の関数の呼び出し終了に伴い前記レジスタへ前記スタックメモリに退避されているデータを伸張せずに復帰する復帰手段とを備えること

を特徴とする請求項 1 記載の情報処理装置。

【請求項 3】 前記退避手段は、

所定の関数の呼び出しに伴いスタックメモリへ前記レジスタの内容を圧縮して退避する場合に、所定の関数の呼び出し終了に伴い前記レジスタへスタックメモリに退避されているデータを伸張して復帰することを示す伸張情報と、前記レジスタの内容を圧縮したデータとを対応付けてスタックメモリに退避する伸張情報退避手段を含み、

前記情報処理装置はさらに、

所定の関数の呼び出し終了に伴い前記レジスタへ復帰しようとするデータと対応付けられている伸張情報が、スタックメモリ中に有るか否かを識別する伸張識別手段と、

伸張識別手段による識別結果が肯定的である場合には所定の関数の呼び出し終了に伴い前記レジスタへスタックメモリ中の前記復帰しようとするデータを伸張して復帰し、伸張識別手段による識別結果が否定的である場合には所定の関数の呼び出し終了に伴い前記レジスタへスタックメモリ中の前記復帰しようとするデータを伸張せずに復帰する復帰手段とを備えること

を特徴とする請求項 1 記載の情報処理装置。

【請求項 4】 前記退避手段はさらに、

前記レジスタに格納されている第 1 データを所定のアルゴリズムに従って、第 2 データに変換するデータ変換手段と、

前記第 2 データのデータサイズを、圧縮効率を示すしきい値と比較する比較手段と、

前記第 2 データのデータサイズがしきい値より小さい場合には前記第 2 データをスタックメモリへ退避し、前記第 2 データのデータサイズがしきい値より大きい場合には前記第 1 データをスタックメモリへ退避する選択退避手段とを含むこと

を特徴とする請求項 3 記載の情報処理装置。

【請求項 5】 前記退避手段は、圧縮識別手段による識別結果が肯定的である場合、前記所定の関数の呼び出し元における呼び出し時にスタックメモリへ前記レジスタの内容を圧縮して退避し、

前記復帰手段は、伸張識別手段による識別結果が肯定的である場合、前記所定の関数の呼び出し元における呼び出し終了時に前記レジスタへスタックメモリに退避されているデータを伸張して復帰すること

を特徴とする請求項 2、及び請求項 3 のいずれか 1 項に記載の情報処理装置。

【請求項 6】 前記退避手段は、圧縮識別手段による識別結果が肯定的である場合、呼び出された前記所定の関数における処理開始時にスタックメモリへ前

記レジスタの内容を圧縮して退避し、

前記復帰手段は、伸張識別手段による識別結果が肯定的である場合、呼び出された前記所定の関数における処理終了時に前記レジスタヘスタックメモリに退避されているデータを伸張して復帰すること

を特徴とする請求項 2、及び請求項 3 のいずれか 1 項に記載の情報処理装置。

【請求項 7】 演算に用いるデータを保持する 1 つ以上のレジスタを有し、機械語プログラムに従って処理を行うプロセッサを含む情報処理装置における情報処理方法であって、

所定の関数の呼び出しに伴いスタックメモリへ前記レジスタの内容を圧縮して退避することを示す情報が、前記機械語プログラム中に有るか否かを識別する圧縮識別ステップと、

圧縮識別ステップによる識別結果が肯定的である場合には所定の関数の呼び出しに伴いスタックメモリへ前記レジスタの内容を圧縮して退避し、圧縮識別ステップによる識別結果が否定的である場合には所定の関数の呼び出しに伴いスタックメモリへ前記レジスタの内容を圧縮せずに退避する退避ステップと

を備えることを特徴とする情報処理方法。

【請求項 8】 プログラム変換装置であって、

1、又は複数の関数を含む入力プログラムを入手する入手手段と、

所定の関数の呼び出しに伴いスタックメモリへプロセッサ内のレジスタの内容を圧縮して退避するか、圧縮せずにそのまま退避するかを前記入力プログラムにおいて判定する判定手段と、

当該判定手段により圧縮して退避すると判定された場合、スタックメモリへ前記レジスタの内容を圧縮して退避することをプロセッサに示す情報を備える出力プログラムに、前記入力プログラムを変換する変換手段と

を備えることを特徴とするプログラム変換装置。

【請求項 9】 前記判定手段は、

前記レジスタの内容を格納したスタックメモリを参照するスタックアクセス関数を検出する検出手段と、

前記スタックアクセス関数、及び前記スタックアクセス関数の呼び出し側の上

位に位置する全ての階層の関数の呼び出しに伴いスタックメモリへ前記レジスタの内容を圧縮せずにそのまま退避すると判定する除外手段とを含むこと

を特徴とする請求項 8 記載のプログラム変換装置。

【請求項 10】 前記判定手段は、

スタックメモリへ前記レジスタの内容を圧縮して退避することを示す情報が前記所定の関数に事前に付加されている場合、前記所定の関数の呼び出しに伴いスタックメモリへ前記レジスタの内容を圧縮して退避すると判定する事前指定判定手段を含むこと

を特徴とする請求項 8 記載のプログラム変換装置。

【請求項 11】 前記判定手段は、

前記所定の関数が内部にサブルーチンを持つ場合、前記入力プログラムが含む関数群の階層構造を示すネスト情報に基づいて、前記所定の関数の呼び出しに伴いスタックメモリへ前記レジスタの内容を圧縮して退避するか、圧縮せずにそのまま退避するかを判定するネスト判定手段を含むこと

を特徴とする請求項 8 記載のプログラム変換装置。

【請求項 12】 前記変換手段は、

前記所定の関数の呼び出しに伴いスタックメモリへ前記レジスタの内容を圧縮して退避することをプロセッサに示す情報を、前記所定の関数の呼び出し元関数に付加する圧縮情報付加手段と、

前記所定の関数の呼び出し終了に伴い前記レジスタへスタックメモリに退避されているデータを伸張し復帰することをプロセッサに示す情報を、前記所定の関数の呼び出し元関数に付加する伸張情報付加手段とを含むこと

を特徴とする請求項 8 記載のプログラム変換装置。

【請求項 13】 前記変換手段は、

前記所定の関数の呼び出しに伴いスタックメモリへ前記レジスタの内容を圧縮して退避することをプロセッサに示す情報を、呼び出される前記所定の関数に付加する圧縮情報付加手段と、

前記所定の関数の呼び出し終了に伴い前記レジスタへスタックメモリに退避されているデータを伸張し復帰することをプロセッサに示す情報を、呼び出される

前記所定の関数に付加する伸張情報付加手段とを含むこと
を特徴とする請求項8記載のプログラム変換装置。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】

本発明は、関数の呼び出しに伴い発生するスタックメモリへのレジスタの内容の退避時にスタックメモリの使用効率を向上させる技術に関する。

【0002】

【従来の技術】

プログラムを実行するコンピュータは、プログラムのメインルーチンから関数の呼び出しに伴い、現在実行している処理のアドレスや、レジスタなどの資源をスタックメモリに退避し、呼び出された関数からの復帰に伴い、スタックメモリからレジスタを復帰し、退避したアドレスへ処理を戻す。しかしながら、全てのレジスタを退避するとそれだけ退避に要するサイクル数やメモリ容量が大きくなりシステム性能が劣化するので、関数の前後で値を保証する必要がある保証レジスタのみを退避の対象として、退避すべきレジスタの数を削減している。

【0003】

サイクル数削減という点では、例えばサブルーチンの呼び出し側と呼び出され側とが異なるファイルに配置されていてもパイプラインの乱れを生じることなくサブルーチンを呼び出すプログラム変換装置及びプロセッサが特開平8-305581号公報に開示されている。

【0004】

【発明が解決しようとする課題】

しかしながら、近年のプログラムの大規模化に伴い、プロセッサが有するレジスタの本数は増大する傾向にある。従って、保証レジスタ群だけをスタックメモリへ退避するとしてもそのデータ量は大量となり、スタックメモリのオーバーフローを回避するために大容量のスタックメモリを必要とする問題がある。

【0005】

本発明はかかる問題に鑑み、スタックメモリの使用効率を上げることでスタック

クメモリのオーバーフローが発生する危険性を抑え、省メモリを実現する情報処理装置、情報処理方法、およびプログラム変換装置を提供することを目的とする。

【0006】

【課題を解決するための手段】

上記目的を達成するために、本発明に係る情報処理装置は、演算に用いるデータを保持する1つ以上のレジスタを有し、機械語プログラムに従って処理を行うプロセッサを含む情報処理装置であって、所定の関数の呼び出しに伴いスタックメモリへ前記レジスタの内容を圧縮して退避することを示す情報が、前記機械語プログラム中に有るか否かを識別する圧縮識別手段と、圧縮識別手段による識別結果が肯定的である場合には所定の関数の呼び出しに伴いスタックメモリへ前記レジスタの内容を圧縮して退避し、圧縮識別手段による識別結果が否定的である場合には所定の関数の呼び出しに伴いスタックメモリへ前記レジスタの内容を圧縮せずに退避する退避手段とを備えることを特徴とする。

【0007】

また、本発明に係る情報処理方法は、演算に用いるデータを保持する1つ以上のレジスタを有し、機械語プログラムに従って処理を行うプロセッサを含む情報処理装置における情報処理方法であって、所定の関数の呼び出しに伴いスタックメモリへ前記レジスタの内容を圧縮して退避することを示す情報が、前記機械語プログラム中に有るか否かを識別する圧縮識別ステップと、圧縮識別ステップによる識別結果が肯定的である場合には所定の関数の呼び出しに伴いスタックメモリへ前記レジスタの内容を圧縮して退避し、圧縮識別ステップによる識別結果が否定的である場合には所定の関数の呼び出しに伴いスタックメモリへ前記レジスタの内容を圧縮せずに退避する退避ステップとを備えることを特徴とする。

【0008】

これによって、関数の呼び出しに伴いスタックメモリへ退避するレジスタの内容を圧縮することができるので、関数の呼び出しの前後でのレジスタの値を保証したうえで、スタックメモリに格納するデータ量を削減することができる。

従って、スタックメモリの使用効率を上げることでスタックメモリのオーバー

フローが発生する危険性を抑え、省メモリを実現することができる。

【0009】

上記目的を達成するために、本発明に係わるプログラム変換装置は、プログラム変換装置であって、1、又は複数の関数を含む入力プログラムを入手する入手手段と、所定の関数の呼び出しに伴いスタックメモリへプロセッサ内のレジスタの内容を圧縮して退避するか、圧縮せずにそのまま退避するかを前記入力プログラムにおいて判定する判定手段と、当該判定手段により圧縮して退避すると判定された場合、スタックメモリへ前記レジスタの内容を圧縮して退避することをプロセッサに示す情報を備える出力プログラムに、前記入力プログラムを変換する変換手段とを備えることを特徴とする。

【0010】

これによって、呼び出しに伴いスタックメモリへ退避するレジスタの内容を圧縮するのに適した関数を、圧縮対象とすることができる。

従って、スタックメモリの使用効率を上げることができるプログラムを実現することができる。

【0011】

【発明の実施の形態】

以下、本発明の実施の形態について、図1から図26を用いて説明する。

<概要>

本発明は、関数の呼び出しに伴いスタックメモリに退避する保証レジスタの内容を圧縮することで、スタックメモリの効率的な使用を実現するものである。

【0012】

以下では、まず、プログラムに従って動作するコンピュータであり、関数の呼び出し、復帰に伴いスタックへ退避、復帰する保証レジスタの内容を圧縮、伸張するかを示す情報を付加された実行ファイル20を生成するプログラム変換装置100と関連ファイル群について説明し、続いて、プログラムに従って動作するコンピュータであり、プログラム変換装置100が生成する実行ファイル20を実行する情報処理装置200について説明する。

【0013】

<構成>

図1は、本発明の実施の形態に係わるプログラム変換装置、及び情報処理装置を備えた情報処理システムの構成を示す図である。

図に示す情報処理システムは、ソースファイル群10を変換し、実行ファイル20を生成するプログラム変換装置100と、実行ファイル20を実行する情報処理装置200とから構成される。

【0014】

(ソースファイル)

ソースファイル群10は、ここではソースファイル11、及びソースファイル12の2つのソースファイルから構成される。

図2は、ソースファイル11、図3はソースファイル12の一例を模式的に示す図である。

【0015】

ソースファイル11、ソースファイル12は高級言語により記述されたプログラムファイルであり、ソースファイル11は関数mainに関する記述部11aと、関数func#aに関する記述部11bと、関数func#bに関する記述部11cと、関数func#dに関する記述部11dと、関数func#eに関する記述部11eと、関数func#fに関する記述部11gとを含み、ソースファイル12は関数func#cに関する記述部12aを含む。ここで記述される関数群は呼び出し、及び復帰により処理を分岐するツリー構造を持つ。

【0016】

関数mainに関する記述部11aは、関数func#aを呼び出す命令と、関数func#bを呼び出す命令と、関数func#eを呼び出す命令とを含む。関数func#aに関する記述部11bは関数func#cを呼び出す命令を含む。関数func#cに関する記述部12aは関数func#fを呼び出す命令を含む。関数func#bに関する記述部11cは関数func#cを呼び出す命令と、関数func#dを呼び出す命令と、前記2つの命令のどちらか一方に条件分岐する判定とを含む。関数func#dに関する記述部11dは、インラインアセンブラ記述された、スタックポインタが指し示すアドレスから4バイト上位のアドレスのメモリ内容をレジスタr0に転送するロード命令と、レジ

スタ r 0 の内容と “0xf0” の論理積をレジスタ r 0 に格納する論理積演算命令と、レジスタ r 0 の内容をスタックポインタから 4 バイト上位のアドレスに転送するストア命令を含む。関数 func#e に関する記述部 1 1 e は、プラグマ #STACK#COMPRESS の記述部 1 1 f を含む。ここでプラグマ #STACK#COMPRESS とは関数の呼び出し時に保証レジスタからスタックメモリへ退避する退避データを圧縮することを、プログラム変換装置 1 0 0 に明示するプラグマである。

【 0 0 1 7 】

(プログラム変換装置)

図 4 は、本発明の実施の形態に係わるプログラム変換装置の構成を示す図である。

プログラム変換装置 1 0 0 は、コンパイル部 1 1 0、中間コード保持部 1 2 0、選択候補情報保持部 1 3 0、リンク部 1 4 0、及び記憶部 1 5 0 から構成される。

【 0 0 1 8 】

コンパイル部 1 1 0 はソースファイル群 1 0 を取得し、中間コード、及び選択候補テーブル 1 3 1 を作成するコンパイラであり、ソースファイル取得部 1 1 1、翻訳部 1 1 2、スタックアクセス検出部 1 1 3、及びユーザ指定関数抽出部 1 1 4 から構成される。

ソースファイル取得部 1 1 1 は、記憶部 1 5 0 が保持するソースファイル 1 1、及びソースファイル 1 2 を取得し、翻訳部 1 1 2、スタックアクセス検出部 1 1 3、及びユーザ指定関数抽出部 1 1 4 にソースファイル 1 1、及びソースファイル 1 2 を渡す。

【 0 0 1 9 】

翻訳部 1 1 2 は、ソースファイル 1 1、及びソースファイル 1 2 に記述される各関数を、呼び出し元の関数から処理を分岐するコール命令、関数内の処理を行う機械語命令、及び呼び出し元の関数へ処理を復帰するリターン命令に変換して、ソースファイル 1 1、及びソースファイル 1 2 から機械語の中間コード群を生成する。翻訳部 1 1 2 は、生成した中間コード群を中間コード保持部 1 2 0 へ出力する。

【0020】

スタックアクセス検出部113は、ソースファイル11、及びソースファイル12に含まれる各関数において各関数内の処理を調査し、各関数の処理開始時のスタックポインタの値より上位のアドレスを参照する記述がある関数を検出する。スタックアクセス検出部113は検出した関数の名前と関数評価値を選択候補情報保持部130が保持する選択候補テーブル131に書き込む。ここで検出された関数の関数評価値は0とされる。

【0021】

ユーザ指定関数抽出部114は、ソースファイル11、及びソースファイル12に含まれる関数のうち、スタックアクセス検出部113によって選択候補テーブル131に書き込まれていない関数においてプラグマ#STACK#COMPRESSの記述の有無を調査する。ユーザ指定関数抽出部114は、ここで調査した関数のうちプラグマ#STACK#COMPRESSが記述された関数の名前と関数評価値を2として選択候補テーブル131に書き込み、また、プラグマ#STACK#COMPRESSが記述されていない関数の名前と関数評価値を1として選択候補テーブル131に書き込む。

【0022】

中間コード保持部120は機械語の中間コード群を保持するメモリ領域である。

選択候補情報保持部130は選択候補テーブル131を保持するメモリ領域である。

図5は、選択候補情報保持部130が保持する選択候補テーブル131のデータ構造を示す図である。

【0023】

選択候補テーブル131はソースファイル群10に含まれる関数の数と同数の選択候補情報から構成される。各選択候補情報は、各関数と1対1に対応し、関数名131a、及び関数評価値131bから構成される。関数評価値131bは、0、1、及び2の何れかの値をとり、スタックアクセス検出部113で検出された、関数の処理開始時のスタックポインタが示すアドレスより、上位のアドレスのスタックメモリにアクセスする処理がある関数、つまり、既にいずれかの関

数の呼び出しによって保証レジスタの値が格納されているスタックメモリを参照する関数の関数評価値 131b は 0、ユーザ指定関数抽出部 114 で検出された、プリAGMA#STACK#COMPRESS が記述された関数の関数評価値 131b は 2、上記 2 種類以外の関数の関数評価値 131b は 1 となる。

【0024】

リンク部 140 は中間コード、及び選択候補テーブル 131 から実行ファイル 20 を作成するリンカであり、連結部 141、関数ツリー情報作成部 142、関数ツリー情報保持部 143、関数選択部 144、及び圧縮・伸張ビット付加部 145 から構成される。

連結部 141 は中間コード保持部 120 に保持される中間コード群に実行時の再配置情報などを追加しリンクすることで、実行可能な機械語コードを生成し圧縮・伸張ビット付加部に出力する。

【0025】

図 6 は、呼び出し関係にある関数のツリー構造を示す図である。各関数から見て、図の横軸の左方向にある呼び出し元の関数をツリー構造の上位の関数、右方向にある呼び出し先の関数をツリー構造の下位の関数と呼ぶ。また、図に示すようにメインルーチンである関数 main からは、関数 func#a、関数 func#b、及び関数 func#e の 3 つのサブルーチンが呼び出され、これら 3 つのサブルーチンの先頭関数をツリー構造の最上位の関数とする。

【0026】

関数ツリー情報作成部 142 は中間コード保持部 120 に保持される中間コード群から、自らより下位の関数を持たない最下位の関数を検出し、最下位の関数から見て最上位の関数までの呼び出し関係をツリー構造として抽出する。ここでは、関数 func#a、関数 func#c、及び関数 func#f が含まれるツリー構造と、関数 func#b、関数 func#c、及び関数 func#f が含まれるツリー構造と、関数 func#b、及び関数 func#d が含まれるツリー構造と、関数 func#e のみのツリー構造とが抽出される。関数ツリー情報作成部 142 は抽出した各ツリー構造についてツリー番号を付し、ツリー構造を構成する各関数の関数評価値 131b の積算値であるツリー評価値を算出する。以上の情報をもとに関数ツリー情報作成部 142 はツリー番

号とツリー評価値を関数ツリー情報保持部 1 4 3 に書き込み、関数ツリー情報テーブル 1 4 3 a を作成する。

【 0 0 2 7 】

図 7 は、関数ツリー情報保持部 1 4 3 で保持する関数ツリー情報テーブル 1 4 3 a のデータ構造を示す図である。

関数ツリー情報テーブル 1 4 3 a は中間コード中の関数のツリー構造の数と同数のツリー情報から構成される。各ツリー情報は各ツリー構造と 1 対 1 で対応し、ツリー番号 1 4 3 b、及びツリー評価値 1 4 3 c から構成される。

【 0 0 2 8 】

ツリー評価値 1 4 3 c はツリー構造を構成する各関数の関数評価値 1 3 1 b を互いに乗じた値である。

ここでは、ソースファイル 1 1、及びソースファイル 1 2 に基づいて作成された関数ツリー情報テーブル 1 4 3 a に、ツリー番号 1 4 3 b が 1、2、3、及び 4 である 4 つのツリー情報が含まれる。

【 0 0 2 9 】

関数選択部 1 4 4 は、ツリー評価値 1 4 3 c が 0 ではないツリー構造、つまりツリー構造を構成する関数の中に、すでにいずれかの関数の呼び出しによって保証レジスタの値が格納されているスタックメモリを参照する関数を含まないツリー構造の最上位の関数、及び関数評価値 1 3 1 b が 2 の関数、つまりユーザ指定を受けた関数を圧縮対象として選択する。

【 0 0 3 0 】

圧縮・伸張ビット付加部 1 4 5 は、連結部 1 4 1 で中間ファイル群をリンクし生成した機械語コードにおいて、関数選択部 1 4 4 で選択した関数のコール命令に保証レジスタの圧縮を示す 1 ビットの圧縮制御ビット、及びリターン命令にスタックの伸張を示す 1 ビットの伸張制御ビットを付加し実行ファイル 2 0 として記憶部 1 5 0 に出力する。

【 0 0 3 1 】

図 8 は、実行ファイル 2 0 の内容を模式的に示す図である。

実行ファイル 2 0 では、関数 func#a へ分岐するコール命令 3 0 1 と、関数 func

#eへ分岐するコール命令302に圧縮制御ビットが付加され、関数func#aから復帰するリターン命令303と、関数func#eから復帰するリターン命令304に伸張制御ビットが付加されている。

【0032】

ここで、コール命令は“CALL label,Regs”のようにニーモニックで表され、保証レジスタRegsの内容をスタックメモリに退避し、labelに処理を分岐する。

1ビットの圧縮制御ビットが付加されたコール命令“CALL label,Regs,stack#compress”は保証レジスタRegsの内容を圧縮してスタックメモリに退避し、処理をlabelに分岐する。

【0033】

リターン命令は“RET Regs”のようにニーモニックで表され、スタックメモリの内容を保証レジスタRegsに復帰し、スタックメモリに格納されたアドレスに処理を分岐する。1ビットの伸張制御ビットが付加されたリターン命令“RET Regs,stack#compress”はスタックメモリの内容を伸張して保証レジスタRegsに復帰し、処理をスタックメモリに格納されたアドレスに分岐する。

【0034】

尚、本実施の形態においては、コール命令、及びリターン命令において退避、復帰すべき保証レジスタを明示したが、関数への分岐命令であるコール命令、及びリターン命令において保証レジスタの退避、復帰を行わず、ストア命令、及びロード命令によって、分岐する前にレジスタを退避、復帰することも可能である。また、分岐後にコーリー側でレジスタの退避、復帰を行うことも可能である。これらの場合、関数の呼び出しと復帰に伴う保証レジスタの退避、復帰時に保証レジスタの内容を圧縮、伸張することを示す1ビットの情報“stack#compress”は、ストア命令、及びロード命令に付加される。

【0035】

図9は、分岐後にコーリー側で保証レジスタの退避、復帰を行う実行ファイルの内容を模式的に示す図である。

図9では、関数func#aの処理開始時に保証レジスタr0～r7の内容をスタックポインタが示すアドレスに転送するストア命令305に圧縮制御ビットが付加

され、関数func#aの処理終了時にスタックポインタが指すメモリの内容を保証レジスタr0～r7に転送するロード命令306に伸張制御ビットが付加される。

【0036】

尚、本実施の形態で退避するレジスタは、保証レジスタであるr0～r7の8個のレジスタであるが、保証レジスタとされるレジスタは8個とは限らず、それ以外の数のレジスタを保証レジスタとしてもよい。

図4に示す記憶部150は、メモリであり、ソースファイル群10、及び上述の構成により生成される実行ファイル20を保持する。

【0037】

尚、本実施の形態におけるプログラム変換装置100は、翻訳部112で変換した機械語の中間コード群を中間コード保持部120で保持するが、機械語の中間コード群を中間ファイルとして記憶部150や、外部の記録媒体に出力する構成としてもよい。

また、本実施の形態におけるプログラム変換装置100は、ソースファイル群10を記憶部150に保持しているが、ソースファイル群10を外部の記録媒体から取得する構成としてもよい。

また、本実施の形態におけるプログラム変換装置100は、実行ファイル20を記憶部150に出力するが、実行ファイル20を外部の記録媒体へ出力する構成としてもよい。

【0038】

(情報処理装置)

図10は、本発明の実施の形態に係わる情報処理装置の構成を示す図である。

情報処理装置200は、命令メモリ210、処理部220、データメモリ230、圧縮部240、CSR250、及び伸張部260から構成される。

命令メモリ210はプログラム変換装置100により生成された圧縮・伸張制御ビット付きの実行ファイル20を格納している。

【0039】

処理部220は実行ファイル20を実行する中央演算装置である。

図11は、処理部220の内部の構成を示す図である。

処理部 220 は、IF 221、DEC 222、制御部 223、実行部 224、レジスタ 225、PC 226、及び SP 227 から構成される。

IF 221 は命令フェッチ部であり、命令メモリ 210 に格納されている実行ファイル 20 から機械語命令をフェッチし、DEC 222 へ渡す。

【0040】

DEC 222 は命令デコーダであり、機械語命令を解析する。DEC 222 は内部に圧縮制御ビット付きコール命令デコード部 222a、伸張制御ビット付きリターン命令デコード部 222b を備え、圧縮制御ビット付きコール命令をデコードした場合は、制御部 223 にレジスタ 225 が保持するデータの圧縮の制御を指示し、伸張制御ビット付きリターン命令をデコードした場合は、制御部 223 にスタックメモリ 231 が保持するデータの伸張の制御を指示する。

【0041】

尚、本実施の形態においては、コール命令、リターン命令に圧縮・伸張制御ビットが付加されている構成であるが、ストア命令、ロード命令に圧縮・伸張制御ビットが付加されている構成でもよい。この場合、DEC 222 は内部に圧縮制御ビット付きコール命令デコード部 222a に替えて圧縮制御ビット付きストア命令デコード部 222c を、伸張制御ビット付きリターン命令デコード部 222b に替えて伸張制御ビット付きロード命令デコード部 222d を備える。

【0042】

図 12 は、圧縮・伸張制御ビットが付加されているストア命令、ロード命令をデコードする DEC 222 の内部の構成を示す図である。

図 11 に示す制御部 223 は DEC 222 で解析された機械語命令に従い、実行部 224、圧縮部 240、伸張部 260 を制御する。

コール命令時に保証レジスタ 225a のデータを圧縮してスタックメモリ 231 へ退避する場合は、先ず PC 226 が保持するデータを、次に圧縮部 240 が保証レジスタ 225a のデータを圧縮したデータを、最後に CSR 250 が保持するデータをスタックメモリ 231 へ格納し、コール先の関数へ処理を分岐するように制御する。

【0043】

リターン命令時にスタックメモリ 2 3 1 のデータを伸張して保証レジスタ 2 2 5 a へ復帰する場合は、先ず C S R 2 5 0 のデータを、次に伸張部 2 6 0 に C S R 2 5 0 が示すサイズの保証レジスタ 2 2 5 a のデータを、最後に P C 2 2 6 のデータをスタックメモリ 2 3 1 から復帰させ、P C 2 2 6 が示す呼び出し元の関数へ処理を分岐するように制御する。

【 0 0 4 4 】

実行部 2 2 4 は制御部 2 2 3 の制御により機械語命令を実行する演算器である。

レジスタ 2 2 5 は 3 2 ビットの記録装置 r 0 ~ r 3 1 の 3 2 個からなり、実行部 2 2 4 が演算対象とするデータを保持する。レジスタ 2 2 5 の一部、r 0 ~ r 7 の 8 個からなる合計 2 5 6 ビットは保証レジスタ 2 2 5 a であり、コール命令時にデータをスタックメモリ 2 3 1 へ退避し、リターン命令時にスタックメモリ 2 3 1 からデータを復帰することで関数の呼び出しの前後で値を保証する。

【 0 0 4 5 】

P C 2 2 6 はプログラムカウンタであり、現在実行している機械語命令が命令メモリ 2 1 0 上のどのアドレスかを示す。

S P 2 2 7 はスタックポインタであり、スタックメモリ 2 3 1 の未使用領域の先頭アドレスを保持する。

図 1 0 に示すデータメモリ 2 3 0 はデータを保持するメモリである。

【 0 0 4 6 】

スタックメモリ 2 3 1 は、関数や割込み時の戻りアドレスや、保証レジスタの値の格納に使われるデータメモリ 2 3 0 上の領域である。図 1 3 は、スタックメモリ 2 3 1 を模式的に示す図である。

ここでは、“0x??”と表記する事で“??”の値が 1 6 進数であり、“0b??”と表記する事で“??”の値が 2 進数でありであることを表すこととする。

【 0 0 4 7 】

縦軸に付された値（“0xffff~0x0000”）はスタックメモリ 2 3 1 上のアドレスである。スタックメモリ 2 3 1 にデータが格納される場合、データはスタックメモリ 2 3 1 の最上位のアドレス“0xffff”から順次アドレスの下位の方へ格

納され、スタックメモリ 231 からデータが取り出される場合、データはスタックメモリ 231 の未使用領域の先頭アドレスの 1 つ前のアドレスから、順次アドレスの上位の方向へ取り出される。また、スタックメモリ 231 へのデータの格納時には格納されるデータのサイズ分だけ SP 227 は保持する値を減らし、スタックメモリ 231 からのデータの取り出し時には取り出されるデータのサイズ分だけ SP 227 は保持する値を増やす。これにより、SP 227 は常にスタックメモリ 231 の未使用領域の先頭アドレスを指し示す。

【0048】

本実施の形態においては、スタックメモリ 231 は関数の引数、及び PC 226、保証レジスタ 225 a、CSR 250 等のデータを格納し、FIL0（先入れ後出し）の動作により関数の呼び出し、復帰に伴う処理の分岐を実現する。

図 14 は、保証レジスタ 225 a のデータを退避したスタックメモリ 231 を模式的に示す図である。

【0049】

圧縮部 240 は制御部 223 の制御により、スタックメモリ 231 へ退避する保証レジスタ 225 a のデータの圧縮を行う。

図 15 は、圧縮部 240 の内部の構成を示す図である。

圧縮部 240 はデータ圧縮部 241、及び圧縮制御部 242 から構成される。

データ圧縮部 241 は保証レジスタ 225 a のデータをハフマン符号化する。

【0050】

図 16 は、ハフマン符号対応表を示す図である。

ハフマン符号は出現率の高い値にビット長の短い符号を割り当て、出現率の低い値にビット長の長い符号を割り当てることで、データ圧縮を行う。

図 17 は、図 14 と同じ保証レジスタ 225 a のデータを、図 16 のハフマン符号対応表を基に圧縮して退避したスタックメモリ 231 を模式的に示す図である。

【0051】

図 15 に示す圧縮制御部 242 は内部に圧縮比較部 243、しきい値保持部 244 を含み、圧縮比較部 243 において、しきい値保持部 244 が保持するしき

い値とデータ圧縮部241がハフマン符号化したデータのデータサイズを比較する。比較の結果、データ圧縮部241がハフマン符号化したデータのデータサイズがしきい値より小さい場合、ハフマン符号化したデータを退避データとしてスタックメモリ231へ格納し、CSR250においてイネーブルビットを1に設定し退避データのデータサイズを格納する。比較の結果、データ圧縮部241がハフマン符号化したデータのデータサイズがしきい値より大きい場合、保証レジスタ225aのデータを退避データとしてスタックメモリ231へ格納し、CSR250においてイネーブルビットを0に設定する。

【0052】

図10に示すCSR250は32ビットの記録装置であり、1ビットのイネーブルビットと、31ビットのデータフィールドから構成される。

図18は、CSR250のビット割付けを示す図である。

横軸に付された値はCSR上のビット位置を表す。ビット[0]はスタックメモリ231に退避するデータが圧縮されたデータであるか否かを示すイネーブルビットであり、圧縮されたデータの場合は1、圧縮されたデータではない場合は0の値をとる。ビット[1]～[31]は圧縮されたデータのデータサイズを格納するデータフィールドであり、イネーブルビットが1の場合は圧縮されたデータのデータサイズを、イネーブルビットが0の場合は不定値を格納する。

【0053】

伸張部260は制御部223の制御により、保証レジスタ225aへ復帰するスタックメモリ231のデータの伸張を行う。

図19は、伸張部260の内部の構成を示す図である。

伸張部260は伸張制御部261、及びデータ伸張部262から構成される。

伸張制御部261は、CSR250のイネーブルビットが1の場合は、スタックメモリ231に格納されている退避データをデータ伸張部262で伸張させて保証レジスタ225aに格納し、CSR250のイネーブルビットが0の場合は、スタックメモリ231に格納されている退避データをそのまま保証レジスタ225aに格納する。

【0054】

データ伸張部 262 は、スタックメモリ 231 に格納されているハフマン符号化データにハフマン符号化の逆変換を施しデータを伸張する。

図 20 は、保証レジスタ 225a とスタックメモリ 231 間のデータ退避、復帰に伴うデータの流れを示す図である。

通常のコール命令時、保証レジスタ 225a のデータはスタックメモリ 231 へ退避され（データフロー D101）、リターン命令時に、スタックメモリ 231 から保証レジスタ 225a へデータが復帰される（データフロー D102）。

【0055】

圧縮制御ビット付きコール命令時、保証レジスタ 225a のデータは圧縮部 240 で圧縮される（データフロー D103）、圧縮したデータのデータサイズがしきい値保持部 244 で保持されるしきい値より大きければ、保証レジスタ 225a のデータがそのまま退避データとして、圧縮したデータのデータサイズがしきい値より小さければ、圧縮したデータが退避データとしてスタックメモリ 231 へ格納される（データフロー D104）。このとき、圧縮結果、圧縮データのサイズが CSR250 へ設定され（データフロー D105）、CSR250 のデータがスタックメモリ 231 へ格納される（データフロー D106）。

【0056】

伸張制御ビット付きリターン命令時は、スタックメモリ 231 から CSR250 へデータが復帰され（データフロー D106）、CSR250 のデータからスタックメモリ 231 に格納されている退避データが圧縮されているか否かの情報と、退避データのデータサイズが伸張部 260 に取得され（データフロー D107）、CSR250 が示すサイズの退避データが伸張部 260 に取得される（データフロー D108）。退避データが圧縮されている場合は退避データを伸張部 260 で伸張したデータが、圧縮されていない場合は退避データがそのまま保証レジスタ 225a へ復帰される（データフロー D109）。

【0057】

尚、本実施の形態においては、退避データの変換アルゴリズムとして、ハフマン符号を用いているが、ラン・レンジ手法等、他の可逆符号方式の圧縮アルゴリズムを用いて退避データを変換してもよい。

<動作>

(プログラム変換装置)

ここで、上述のように構成されたプログラム変換装置100の動作について説明する。

【0058】

図21は、プログラム変換装置100の動作の流れを示す図である。

まず、コンパイル部110においてソースファイル群10は機械語の中間コード群に変換され(ステップS1001)、次に選択候補テーブル131が作成される(ステップS1002)。

続いて、リンク部140において中間コード群がリンクされ実行可能な機械語コードが生成され(ステップS1003)、関数選択部144において圧縮対象となる関数が選択される(ステップS1004)。最後に、圧縮・伸張ビット付加部が機械語コード中の、圧縮対象の関数のコール命令に圧縮制御ビット、及びリターン命令に伸張制御ビットを付加し実行ファイルとして出力する(ステップS1005)。

【0059】

次に、プログラム変換装置100の動作中で選択候補テーブルを作成する動作(ステップS1002)を、図を用いて説明する。

図22は、コンパイル部110において選択候補テーブルを作成する動作の流れを示す図である。

まず、ある関数を調査対象とし調査を開始する(ステップS1101)。スタックアクセス検出部113が、対象の関数内で関数先頭でのスタックポインタが示すアドレスより上位のアドレスのスタックメモリへのアクセスがあるか否かを判定する(ステップS1102)。スタックアクセス検出部113が、対象の関数内でアクセスがあると判定する場合(ステップS1102:Yes)、対象の関数の関数評価値131bに0を設定する(ステップS1103)。

【0060】

一方、スタックアクセス検出部113が、対象の関数内でアクセスがないと判定する場合(ステップS1102:No)、ユーザ指定関数抽出部114が、対象

の関数内にプラグマ#STACK#COMPRESSが記述されているか否かを判定する（ステップS 1 1 0 4）。ユーザ指定関数抽出部 1 1 4 が、対象の関数内にプラグマ#STACK#COMPRESSが記述されていると判定する場合（ステップS 1 1 0 4 : Yes）、対象の関数の関数評価値 1 3 1 b に 2 を設定し（ステップS 1 1 0 5）、ユーザ指定関数抽出部 1 1 4 が、対象の関数内にプラグマ#STACK#COMPRESSが記述されていないと判定する場合（ステップS 1 1 0 4 : No）、対象の関数の関数評価値 1 3 1 b に 1 を設定する（ステップS 1 1 0 6）。

【 0 0 6 1 】

上記動作で対象の関数の関数評価値 1 3 1 b が設定された後、さらに未調査の関数が存在する場合（ステップS 1 1 0 7 : Yes）、未調査の関数を新たな調査対象とし調査を開始する（ステップS 1 1 0 1）。未調査の関数が存在しない場合（ステップS 1 1 0 7 : No）、選択候補テーブルの作成が終了する。

次に、プログラム変換装置 1 0 0 の動作中で圧縮対象となる関数を選択する動作（ステップS 1 0 0 4）を、図を用いて説明する。

【 0 0 6 2 】

図 2 3 は、リンク部 1 4 0 において圧縮対象となる関数を選択する動作の流れを示す図である。

まず、ある関数を調査対象とし調査を開始する（ステップS 1 2 0 1）。関数選択部 1 4 4 が、対象の関数の関数評価値 1 3 1 b に 2 が設定されているか否かを判定する（ステップS 1 2 0 2）。関数選択部 1 4 4 が、対象の関数の関数評価値 1 3 1 b に 2 が設定されていると判定する場合（ステップS 1 2 0 2 : Yes）、対象の関数が圧縮対象となる関数に選択される（ステップS 1 2 0 7）。

【 0 0 6 3 】

一方、関数選択部 1 4 4 が、対象の関数の関数評価値 1 3 1 b に 2 が設定されていないと判定する場合（ステップS 1 2 0 2 : No）、対象の関数がツリーの先頭関数となるツリーがあるか否かを判定する（ステップS 1 2 0 3）、対象の関数がツリーの先頭関数となるツリーがある場合（ステップS 1 2 0 3 : Yes）、対象の関数がツリーの先頭関数となるツリーを調査対象とし（ステップS 1 2 0 4）、調査対象のツリーのツリー評価値 1 4 3 c に、0 が設定されているか否か

を判定する（ステップ S1205）。調査対象のツリーのツリー評価値 143c に 0 が設定されていない場合（ステップ S1205：No）、対象の関数がツリーの先頭関数となる未調査のツリーが他にあるか否かを判定する（ステップ S1206）。対象の関数がツリーの先頭関数となる未調査のツリーが他にある場合（ステップ S1206：Yes）、未調査のツリーを調査対象とする（ステップ S1204）。対象の関数がツリーの先頭関数となる未調査のツリーが他にない場合（ステップ S1206：No）、対象の関数が圧縮対象となる関数に選択される（ステップ S1207）。

【0064】

対象の関数がツリーの先頭関数となるツリーがない場合（ステップ S1203：No）、及び調査対象のツリーのツリー評価値 143c に、0 が設定されている場合（ステップ S1205：Yes）、または上記動作で対象の関数が圧縮対象となる関数に選択された後、さらに未調査の関数が存在する場合（ステップ S1208：Yes）、未調査の関数を新たな調査対象とし調査を開始する（ステップ S1201）。未調査の関数が存在しない場合（ステップ S1208：No）、圧縮対象となる関数の選択が終了する。

（情報処理装置）

ここで、上述のように構成された情報処理装置 200 の動作について説明する。

【0065】

情報処理装置 200 は、処理を複数のステージに分けるパイプラインを採用することにより機械語命令を高速に実行するが、ここでは一つの機械語命令について処理の各ステージを通した動作を説明する。

図 24 は、情報処理装置 200 の動作の流れを示す図である。

まず、IF221 が、PC226 が示す命令メモリ 210 上のアドレスから機械語命令をフェッチする（ステップ S2001）。次に、DEC222 が、機械語命令の種類を判定する（ステップ S2002）。

【0066】

DEC 222 が、機械語命令をコール命令であると判定する場合（ステップ S 2002 : CALL）、関数の呼び出しと、それに伴う保証レジスタ 225 a の退避が実行される（ステップ S 2003）。

DEC 222 が、機械語命令をリターン命令であると判定する場合（ステップ S 2002 : RET）、関数からの復帰と、それに伴う保証レジスタ 225 a の復帰が実行される（ステップ S 2004）。

【0067】

DEC 222 が、機械語命令をコール命令、及びリターン命令以外であると判定する場合（ステップ S 2002 : else）、実行部 224 が機械語命令を実行する（ステップ S 2005）。

以上の動作により、一つの機械語命令の処理が終了する。

次に、情報処理装置 200 の動作中で関数の呼び出しと、それに伴い保証レジスタ 225 a を退避する動作（ステップ S 2003）を、図を用いて説明する。

【0068】

図 25 は、関数の呼び出しと、それに伴い保証レジスタ 225 a を退避する動作の流れを示す図である。

先ず、DEC 222 が、コール命令に圧縮制御ビットが付加されているか否かを判定する（ステップ S 2101）。DEC 222 が、コール命令に圧縮制御ビットが付加されていないと判定する場合（ステップ S 2101 : No）、制御部 223 が、PC 226 のデータをスタックメモリ 231 に格納し（ステップ S 2102）、続いて、制御部 223 が、保証レジスタ 225 a の 256 ビットのデータをスタックメモリ 231 へ格納する（ステップ S 2103）。

【0069】

一方、DEC 222 が、コール命令に圧縮制御ビットが付加されていると判定する場合（ステップ S 2101 : Yes）、制御部 223 が、PC 226 のデータをスタックメモリ 231 に格納し（ステップ S 2104）、続いて、データ圧縮部 241 が、保証レジスタ 225 a のデータをハフマン符号化し（ステップ S 2105）、圧縮比較部 243 が、ハフマン符号化したデータのサイズがしきい値より小さいか否かを判定する（ステップ S 2106）、圧縮比較部 243 が、ハ

フマン符号化したデータのサイズがしきい値より小さいと判定する場合（ステップ S 2106 : Yse）、圧縮制御部 242 が、ハフマン符号化したデータをスタックメモリ 231 へ格納し（ステップ S 2107）、続いて、圧縮制御部 242 が、CSR 250 のイネーブルビットに 1 を、データフィールドにハフマン符号化したデータのサイズを設定する（ステップ S 2108）。

【0070】

圧縮比較部 243 が、ハフマン符号化したデータのサイズがしきい値より小さくないと判定する場合（ステップ S 2106 : No）、圧縮制御部 242 が、保証レジスタ 225a の 256 ビットのデータをそのままスタックメモリ 231 へ格納し（ステップ S 2109）、続いて、圧縮制御部 242 が、CSR 250 のイネーブルビットに 0 を設定する（ステップ S 2110）。

【0071】

CSR 250 の設定後、圧縮制御部 242 が、CSR 250 のデータをスタックメモリ 231 へ格納する（ステップ S 2111）。

最後に、呼び出し先の関数へ処理を分岐する（ステップ S 2112）。

次に、情報処理装置 200 の動作中で関数からの復帰と、それに伴い保証レジスタ 225a を復帰する動作（ステップ S 2104）を、図を用いて説明する。

【0072】

図 26 は、関数からの復帰と、それに伴い保証レジスタ 225a を復帰する動作の流れを示す図である。

まず、DEC 222 が、リターン命令に伸張制御ビットが付加されているか否かを判定する（ステップ S 2201）。DEC 222 が、リターン命令に伸張制御ビットが付加されていないと判定する場合（ステップ S 2201 : No）、制御部 223 が、スタックメモリ 231 から保証レジスタ 225a へ 256 ビットの退避データを読み出す（ステップ S 2202）。

【0073】

一方、DEC 222 が、リターン命令に伸張制御ビットが付加されていると判定する場合（ステップ S 2201 : Yes）、伸張制御部 261 が、スタックメモリ 231 から CSR 250 へ退避データを読み出し（ステップ S 2203）、続

いて、伸張制御部 261 が、読み出した CSR 250 のイネーブルビットが 1 であるか否かを判定する（ステップ S 2204）。伸張制御部 261 が、読み出した CSR 250 のイネーブルビットが 1 でないと判定する場合（ステップ S 2204 : No）、伸張制御部 261 が、スタックメモリ 231 から保証レジスタ 225 a へ 256 ビットの退避データをそのまま読み出す（ステップ S 2202）。

【0074】

伸張制御部 261 が、読み出した CSR 250 のイネーブルビットが 1 であると判定する場合（ステップ S 2204 : Yes）、伸張制御部 261 が、スタックメモリ 231 から CSR 250 のデータフィールドが示すサイズのハフマン符号化したデータを読み出し（ステップ S 2205）、データ伸張部 262 が、ハフマン符号化したデータを逆変換しデータを伸張し（ステップ S 2206）、伸張制御部 261 が、保証レジスタ 225 a へ 256 ビットに伸張されたデータを復帰する（ステップ S 2207）。

【0075】

以上の動作により保証レジスタ 225 a のデータが復帰した後、制御部 223 が、スタックメモリ 231 から PC 226 へ退避データを読み出し（ステップ S 2208）、最後に、PC 226 が示す呼び出し元の関数へ処理を分岐する（ステップ S 2209）。

<まとめ>

本発明の実施の形態は、関数の呼び出し時にスタックメモリへ退避する保証レジスタの内容を圧縮しスタックメモリの使用効率を上げることにより、スタックメモリのオーバーフローが発生する危険性を抑え、省メモリを実現するものであり、全ての関数の呼び出し時にスタックメモリへ退避するレジスタの内容を圧縮した場合、スタックメモリの使用効率は上がるが関数の呼び出しのたびに圧縮動作が発生しサイクル数が増加することになるので、スタックメモリに格納される期間が長い関数、例えばツリー構造になった関数群の先頭の関数、及びユーザにより指定された関数を呼び出し時にレジスタ内容を圧縮する対象に選択することでサイクル数の増加を抑えつつスタックメモリの使用効率を上げることを実現している。

【0076】

【発明の効果】

本発明に係る情報処理装置は、演算に用いるデータを保持する1つ以上のレジスタを有し、機械語プログラムに従って処理を行うプロセッサを含む情報処理装置であって、所定の関数の呼び出しに伴いスタックメモリへ前記レジスタの内容を圧縮して退避することを示す情報が、前記機械語プログラム中に有るか否かを識別する圧縮識別手段と、圧縮識別手段による識別結果が肯定的である場合には所定の関数の呼び出しに伴いスタックメモリへ前記レジスタの内容を圧縮して退避し、圧縮識別手段による識別結果が否定的である場合には所定の関数の呼び出しに伴いスタックメモリへ前記レジスタの内容を圧縮せずに退避する退避手段とを備えることを特徴とする。

【0077】

これによって、関数の呼び出しに伴いスタックメモリへ退避するレジスタの内容を圧縮することができるので、スタックメモリに格納するデータ量を削減することができる。

従って、スタックメモリの使用効率を上げることでスタックメモリのオーバーフローが発生する危険性を抑え、省メモリを実現することができる。

【0078】

また、前記情報処理装置はさらに、所定の関数の呼び出し終了に伴い前記レジスタへスタックメモリに退避されているデータを伸張して復帰することを示す情報が、前記機械語プログラム中に有るか否かを識別する伸張識別手段と、伸張識別手段による識別結果が肯定的である場合には所定の関数の呼び出し終了に伴い前記レジスタへ前記スタックメモリに退避されているデータを伸張して復帰し、伸張識別手段による識別結果が否定的である場合には所定の関数の呼び出し終了に伴い前記レジスタへ前記スタックメモリに退避されているデータを伸張せずに復帰する復帰手段とを備えることを特徴とする。

【0079】

あるいは、前記退避手段は、所定の関数の呼び出しに伴いスタックメモリへ前記レジスタの内容を圧縮して退避する場合に、所定の関数の呼び出し終了に伴い

前記レジスタヘスタックメモリに退避されているデータを伸張して復帰することを示す伸張情報と、前記レジスタの内容を圧縮したデータとを対応付けてスタックメモリに退避する伸張情報退避手段を含み、前記情報処理装置はさらに、所定の関数の呼び出し終了に伴い前記レジスタへ復帰しようとするデータと対応付けられている伸張情報が、スタックメモリ中に有るか否かを識別する伸張識別手段と、伸張識別手段による識別結果が肯定的である場合には所定の関数の呼び出し終了に伴い前記レジスタヘスタックメモリ中の前記復帰しようとするデータを伸張して復帰し、伸張識別手段による識別結果が否定的である場合には所定の関数の呼び出し終了に伴い前記レジスタヘスタックメモリ中の前記復帰しようとするデータを伸張せずに復帰する復帰手段とを備えることを特徴とする。

【0080】

これによって、関数の呼び出し終了に伴いレジスタへ復帰するスタックメモリ中の圧縮された内容を伸張することができるので、関数の呼び出しの前後でのレジスタの値を保証したうえで、スタックメモリに格納するデータ量を削減することができる。

従って、スタックメモリの使用効率を上げることでスタックメモリのオーバーフローが発生する危険性を抑え、省メモリを実現することができる。

【0081】

また、前記退避手段はさらに、前記レジスタに格納されている第1データを所定のアルゴリズムに従って、第2データに変換するデータ変換手段と、前記第2データのデータサイズを、圧縮効率を示すしきい値と比較する比較手段と、前記第2データのデータサイズがしきい値より小さい場合には前記第2データをスタックメモリへ退避し、前記第2データのデータサイズがしきい値より大きい場合には前記第1データをスタックメモリへ退避する選択退避手段とを含むことを特徴とする。

【0082】

これによって、保証レジスタのデータに圧縮を施し、圧縮効果が有る場合のみスタックメモリに圧縮したデータを退避し、圧縮効果がない場合は保証レジスタのデータをそのままスタックメモリに退避することができる。

従って、保証レジスタの各ビットのデータに相関が少なく、圧縮効率の悪い場合に、ハフマン符号化等のデータ変換によるデータサイズの増大の影響を受けることなくスタックメモリの使用効率を上げることを実現できる。

【0083】

また、前記退避手段は、圧縮識別手段による識別結果が肯定的である場合、前記所定の関数の呼び出し元における呼び出し時にスタックメモリへ前記レジスタの内容を圧縮して退避し、前記復帰手段は、伸張識別手段による識別結果が肯定的である場合、前記所定の関数の呼び出し元における呼び出し終了時に前記レジスタへスタックメモリに退避されているデータを伸張して復帰することを特徴とする。

【0084】

これによって、コール命令の動作時に、スタックメモリへ退避する保証レジスタの内容を圧縮し、リターン命令の動作時に、保証レジスタへ復帰するスタックメモリの内容を伸張することができる。

従って、関数への分岐命令がスタックメモリへの保証レジスタの退避を伴う情報処理装置において、関数の前後でのレジスタの値を保証したうえでスタックメモリに格納するデータ量を削減し、スタックメモリの使用効率を上げることができる。

【0085】

また、前記退避手段は、圧縮識別手段による識別結果が肯定的である場合、呼び出された前記所定の関数における処理開始時にスタックメモリへ前記レジスタの内容を圧縮して退避し、前記復帰手段は、伸張識別手段による識別結果が肯定的である場合、呼び出された前記所定の関数における処理終了時に前記レジスタへスタックメモリに退避されているデータを伸張して復帰することを特徴とする。

【0086】

これによって、ストア命令による保証レジスタのスタックメモリへの退避時に、保証レジスタの内容を圧縮し、ロード命令によるスタックメモリの保証レジスタへの復帰時に、スタックメモリの内容を伸張することができる。

従って、関数への分岐命令がスタックメモリへの保証レジスタの退避を伴わない情報処理装置において、関数の前後でのレジスタの値を保証したうえでスタックメモリに格納するデータ量を削減し、スタックメモリの使用効率を上げることができる。

【0087】

また、演算に用いるデータを保持する1つ以上のレジスタを有し、機械語プログラムに従って処理を行うプロセッサを含む情報処理装置における情報処理方法であって、所定の関数の呼び出しに伴いスタックメモリへ前記レジスタの内容を圧縮して退避することを示す情報が、前記機械語プログラム中に有るか否かを識別する圧縮識別ステップと、圧縮識別ステップによる識別結果が肯定的である場合には所定の関数の呼び出しに伴いスタックメモリへ前記レジスタの内容を圧縮して退避し、圧縮識別ステップによる識別結果が否定的である場合には所定の関数の呼び出しに伴いスタックメモリへ前記レジスタの内容を圧縮せずに退避する退避ステップとを備えることを特徴とする。

【0088】

これによって、関数の呼び出しに伴いスタックメモリへ退避するレジスタの内容を圧縮することができるので、スタックメモリに格納するデータ量を削減することができる。

従って、スタックメモリの使用効率を上げることでスタックメモリのオーバーフローが発生する危険性を抑え、省メモリを実現することができる。

【0089】

また、プログラム変換装置であって、1、又は複数の関数を含む入力プログラムを入手する入手手段と、所定の関数の呼び出しに伴いスタックメモリへプロセッサ内のレジスタの内容を圧縮して退避するか、圧縮せずにそのまま退避するかを前記入力プログラムにおいて判定する判定手段と、当該判定手段により圧縮して退避すると判定された場合、スタックメモリへ前記レジスタの内容を圧縮して退避することをプロセッサに示す情報を備える出力プログラムに、前記入力プログラムを変換する変換手段とを備えることを特徴とする。

【0090】

これによって、全ての関数の呼び出しを保証レジスタの圧縮対象とせず、保証レジスタを圧縮する関数を選択的に決定することができる。

従って、全ての関数で保証レジスタを圧縮することによるサイクル数の大幅な増加をさけつつ、スタックメモリの使用効率を上げることを実現することができる。

【0091】

また、前記判定手段は、前記レジスタの内容を格納したスタックメモリを参照するスタックアクセス関数を検出する検出手段と、前記スタックアクセス関数、及び前記スタックアクセス関数の呼び出し側の上位に位置する全ての階層の関数の呼び出しに伴いスタックメモリへ前記レジスタの内容を圧縮せずにそのまま退避すると判定する除外手段とを含むことを特徴とする。

【0092】

これによって、関数の処理開始時のスタックポインタが示すアドレスより上位のアドレスのスタックメモリにアクセスする関数を、ツリー構造の下位に持つ関数を保証レジスタの内容を圧縮する対象から除くことができる。

従って、すでにスタックメモリに格納されているデータを参照する場合に、参照するデータが圧縮されていることを避けることができる。

【0093】

また、前記判定手段は、スタックメモリへ前記レジスタの内容を圧縮して退避することを示す情報が前記所定の関数に事前に付加されている場合、前記所定の関数の呼び出しに伴いスタックメモリへ前記レジスタの内容を圧縮して退避すると判定する事前指定判定手段を含むことを特徴とする。

これによって、変換前のプログラム中で事前に指定された関数を、変換後のプログラムにおいて、保証レジスタの圧縮対象とすることができる。

【0094】

従って、プログラム時（コンパイル前）に、プログラマ（ユーザ）が所望の関数を指定し、保証レジスタの圧縮対象を静的に決定することができる。

また、前記判定手段は、前記所定の関数が内部にサブルーチンを持つ場合、前記入力プログラムが含む関数群の階層構造を示すネスト情報に基づいて、前記所

定の関数の呼び出しに伴いスタックメモリへ前記レジスタの内容を圧縮して退避するか、圧縮せずにそのまま退避するかを判定するネスト判定手段を含むことを特徴とする。

【0095】

これによって、ツリー構造をもつ関数群の任意の関数をコンパイル時に動的に保証レジスタの圧縮対象とすることができる。

従って、一度スタックメモリに退避すると、長期間スタックに留まるツリー構造の上位の関数を、保証レジスタの圧縮対象とすることで、スタックメモリの使用効率を上げることが実現できる。

【0096】

また、前記変換手段は、前記所定の関数の呼び出しに伴いスタックメモリへ前記レジスタの内容を圧縮して退避することをプロセッサに示す情報を、前記所定の関数の呼び出し元関数に付加する圧縮情報付加手段と、前記所定の関数の呼び出し終了に伴い前記レジスタへスタックメモリに退避されているデータを伸張し復帰することをプロセッサに示す情報を、前記所定の関数の呼び出し元関数に付加する伸張情報付加手段とを含むことを特徴とする。

【0097】

これによって、コール命令の動作時にスタックメモリへ退避する保証レジスタの内容を圧縮し、リターン命令の動作時に保証レジスタへ復帰するスタックメモリの内容を伸張することを示す情報をプログラム中に付加することができる。

従って、関数への分岐命令がスタックメモリへの保証レジスタの退避を伴う情報処理装置において、関数の前後でのレジスタの値を保証したうえでスタックメモリに格納するデータ量を削減し、スタックメモリの使用効率を上げるプログラムを生成することができる。

【0098】

また、前記変換手段は、前記所定の関数の呼び出しに伴いスタックメモリへ前記レジスタの内容を圧縮して退避することをプロセッサに示す情報を、呼び出される前記所定の関数に付加する圧縮情報付加手段と、前記所定の関数の呼び出し終了に伴い前記レジスタへスタックメモリに退避されているデータを伸張し復帰

することをプロセッサに示す情報を、呼び出される前記所定の関数に付加する伸張情報付加手段とを含むことを特徴とする。

【0099】

これによって、ストア命令による保証レジスタのスタックメモリへの退避時に保証レジスタの内容を圧縮し、ロード命令によるスタックメモリの保証レジスタへの復帰時にスタックメモリの内容を伸張することを示す情報をプログラム中に付加することができる。

従って、関数への分岐命令がスタックメモリへの保証レジスタの退避を伴わない情報処理装置において、関数の前後でのレジスタの値を保証したうえでスタックメモリに格納するデータ量を削減し、スタックメモリの使用効率を上げるプログラムを生成することができる。

【図面の簡単な説明】

【図1】

本発明の実施の形態に係わるプログラム変換装置、及び情報処理装置を備えた情報処理システムの構成を示す図である。

【図2】

ソースファイル11の一例を模式的に示す図である。

【図3】

ソースファイル12の一例を模式的に示す図である。

【図4】

本発明の実施の形態に係わるプログラム変換装置の構成を示す図である。

【図5】

選択候補情報保持部130が保持する選択候補テーブル131のデータ構造を示す図である。

【図6】

呼び出し関係にある関数のツリー構造を示す図である。

【図7】

関数ツリー情報保持部143で保持する関数ツリー情報テーブル143aのデータ構造を示す図である。

【図 8】

実行ファイル 20 の内容を模式的に示す図である。

【図 9】

分岐後にコーリー側でレジスタの退避、復帰を行う実行ファイルの内容を模式的に示す図である。

【図 10】

本発明の実施の形態に係わる情報処理装置の構成を示す図である。

【図 11】

処理部 220 の内部の構成を示す図である。

【図 12】

圧縮・伸張制御ビットが付加されているストア命令、ロード命令をデコードする DEC 222 の内部の構成を示す図である。

【図 13】

スタックメモリ 231 を模式的に示す図である。

【図 14】

保証レジスタ 225a のデータを退避したスタックメモリ 231 を模式的に示す図である。

【図 15】

圧縮部 240 の内部の構成を示す図である。

【図 16】

ハフマン符号対応表を示す図である。

【図 17】

図 14 と同じ保証レジスタ 225a のデータを、図 16 のハフマン符号対応表を基に圧縮して退避したスタックメモリ 231 を模式的に示す図である。

【図 18】

CSR 250 のビット割付けを示す図である。

【図 19】

伸張部 260 の内部の構成を示す図である。

【図 20】

保証レジスタ 225 a とスタックメモリ 231 間のデータ退避、復帰に伴うデータの流れを示す図である。

【図 21】

プログラム変換装置 100 の動作の流れを示す図である。

【図 22】

コンパイル部 110 において選択候補テーブルを作成する動作の流れを示す図である。

【図 23】

リンク部 140 において圧縮対象となる関数を選択する動作の流れを示す図である。

【図 24】

情報処理装置 200 の動作の流れを示す図である。

【図 25】

関数の呼び出しと、それに伴い保証レジスタ 225 a を退避する動作の流れを示す図である。

【図 26】

関数からの復帰と、それに伴い保証レジスタ 225 a を復帰する動作の流れを示す図である。

【符号の説明】

- 10 ソースファイル群
- 11 ソースファイル
- 11 a 関数mainに関する記述部
- 11 b 関数func#aに関する記述部
- 11 c 関数func#bに関する記述部
- 11 d 関数func#dに関する記述部
- 11 e 関数func#eに関する記述部
- 11 f プラグマ#STACK#COMPRESSの記述部
- 11 g 関数func#fに関する記述部
- 12 ソースファイル

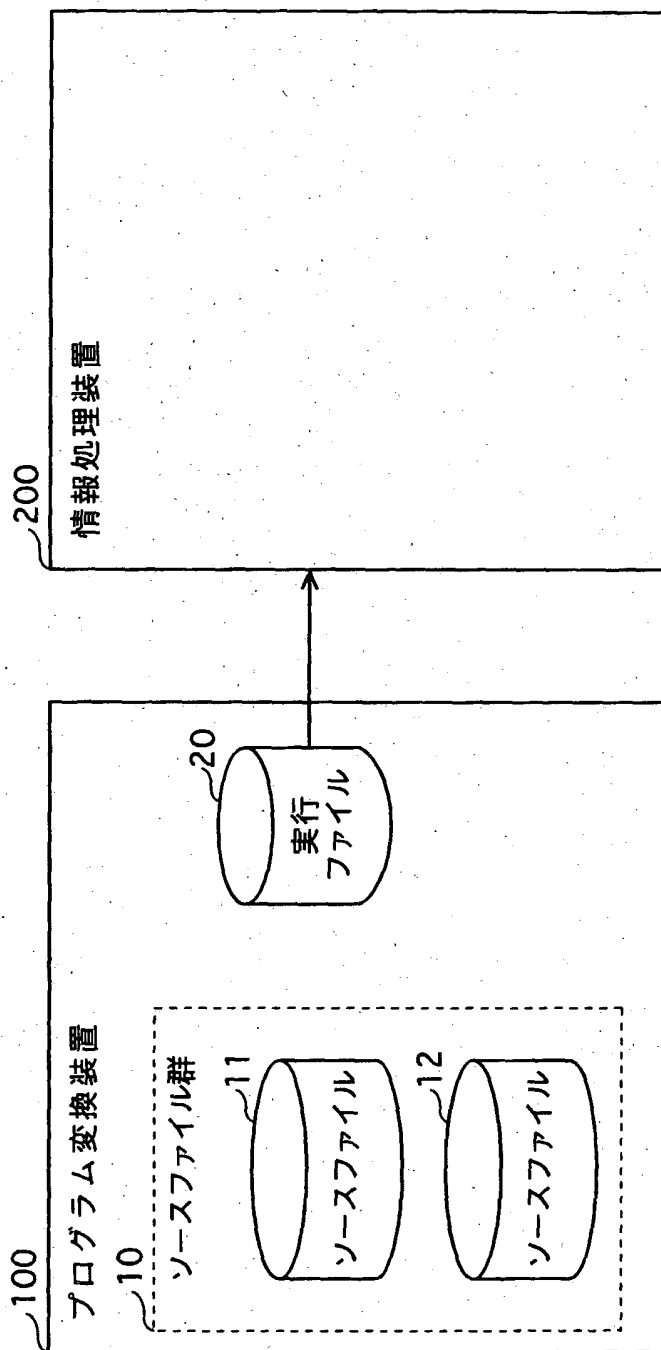
- 1 2 a 関数func#cに関する記述部
- 2 0 実行ファイル
- 1 0 0 プログラム変換装置
- 1 1 0 コンパイル部
- 1 1 1 ソースファイル取得部
- 1 1 2 翻訳部
- 1 1 3 スタックアクセス検出部
- 1 1 4 ユーザ指定関数抽出部
- 1 2 0 中間コード保持部
- 1 3 0 選択候補情報保持部
- 1 3 1 選択候補テーブル
- 1 3 1 a 関数名
- 1 3 1 b 関数評価値
- 1 4 0 リンク部
- 1 4 1 連結部
- 1 4 2 関数ツリー情報作成部
- 1 4 3 関数ツリー情報保持部
- 1 4 3 a 関数ツリー情報テーブル
- 1 4 3 b ツリー番号
- 1 4 3 c ツリー評価値
- 1 4 4 関数選択部
- 1 4 5 圧縮・伸張ビット付加部
- 1 5 0 記憶部
- 2 0 0 情報処理装置
- 2 1 0 命令メモリ
- 2 2 0 処理部
- 2 2 1 I F
- 2 2 2 D E C
- 2 2 2 a 圧縮制御ビット付きコール命令デコード部

- 2 2 2 b 伸張制御ビット付きリターン命令デコード部
- 2 2 2 c 圧縮制御ビット付きストア命令デコード部
- 2 2 2 d 伸張制御ビット付きロード命令デコード部
- 2 2 3 制御部
- 2 2 4 実行部
- 2 2 5 レジスタ
- 2 2 5 a 保証レジスタ
- 2 2 6 P C
- 2 2 7 S P
- 2 3 0 データメモリ
- 2 3 1 スタックメモリ
- 2 4 0 圧縮部
- 2 4 1 データ圧縮部
- 2 4 2 圧縮制御部
- 2 4 3 圧縮比較部
- 2 4 4 しきい値保持部
- 2 5 0 C S R
- 2 6 0 伸張部
- 2 6 1 伸張制御部
- 2 6 2 データ伸張部
- 3 0 1 関数func#aへ分岐するコール命令
- 3 0 2 関数func#eへ分岐するコール命令
- 3 0 3 関数func#aから復帰するリターン命令
- 3 0 4 関数func#eから復帰するリターン命令
- 3 0 5 関数func#aの処理開始時におけるストア命令
- 3 0 6 関数func#aの処理終了時におけるロード命令

【書類名】

図面

【図1】



【図2】

```
11a  main()
    {
        func_a();
        func_b();
        func_e();
    }

11b  func_a()
    {
        func_c();
    }

11c  func_b()
    {
        if(X){
            func_c();
        }else{
            func_d();
        }
    }

11d  func_d()
    {
        |
        asm("load r0,(4,sp)");
        asm("and r0,0xf0");
        asm("store (4,sp),r0");
        |
    }

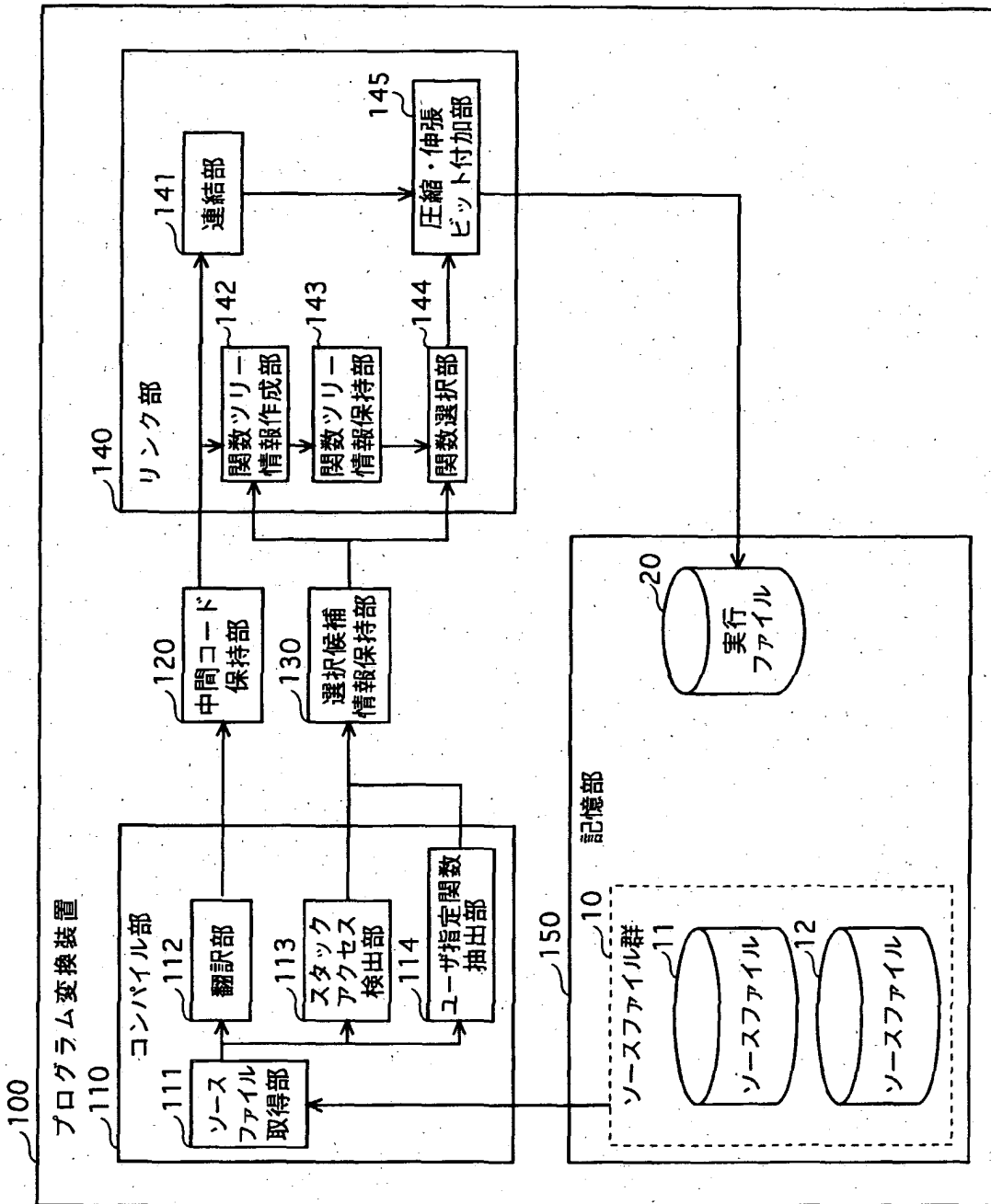
11e  func_e()
    {
11f  #STACK_COMPRESS
    }

11g  func_f()
    {
    }
```


【図3】

12a { func_c()
 {
 func_f();
 }

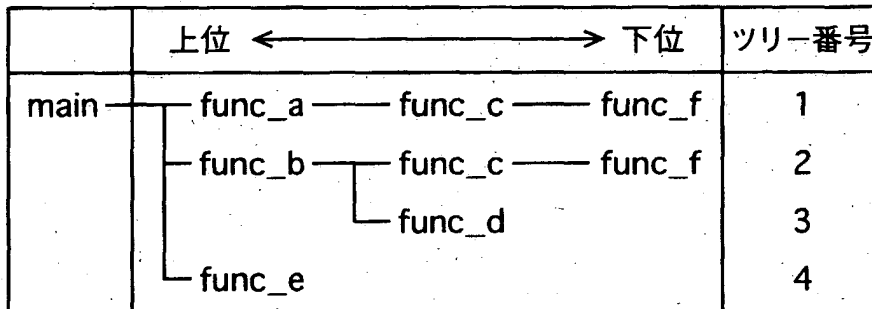
【図4】



【図 5】

関 数 名	関 数 評 価 値
func_a	1
func_b	1
func_c	1
func_d	0
func_e	2
func_f	1

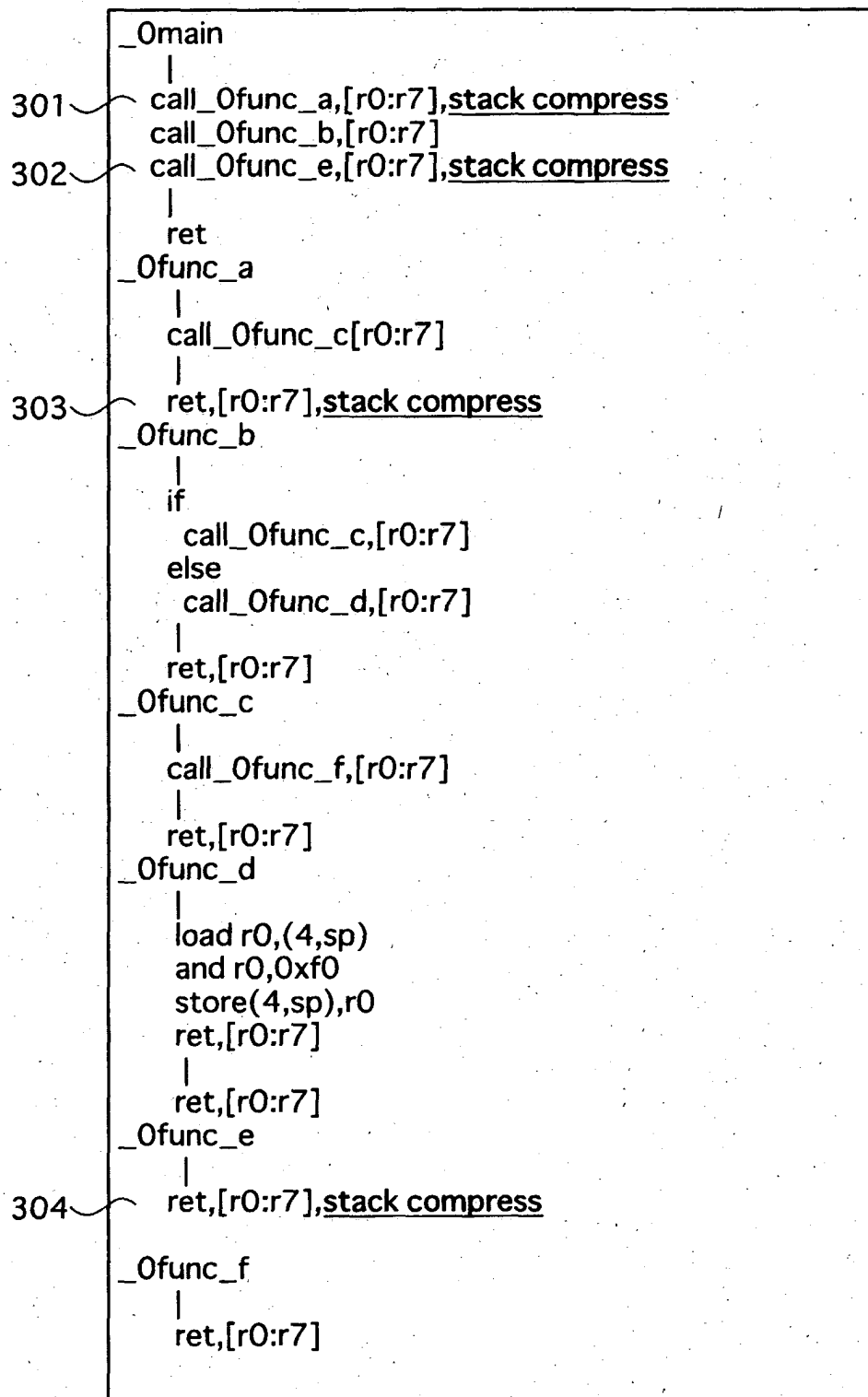
【図 6】



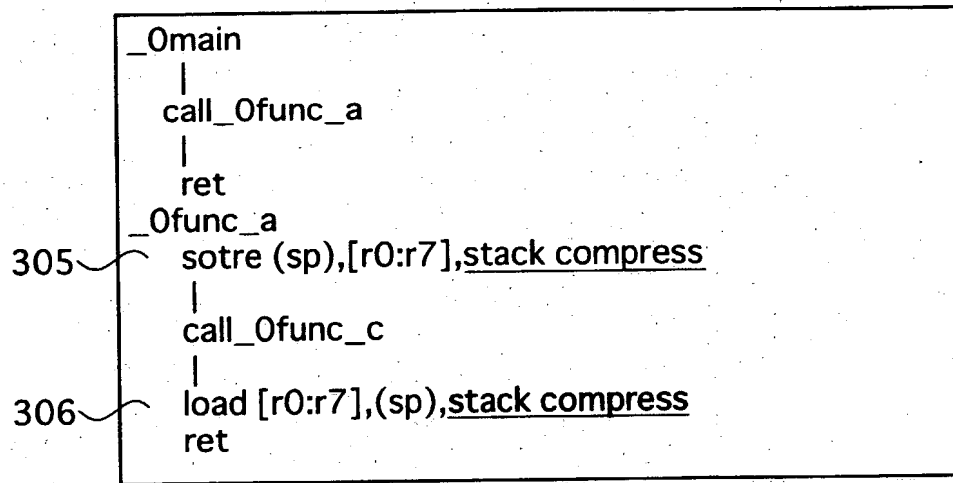
【図 7】

ツリー番号	ツリー評価値
1	1 (=1×1×1)
2	1 (=1×1×1)
3	0 (=1×0)
4	2 (=2)

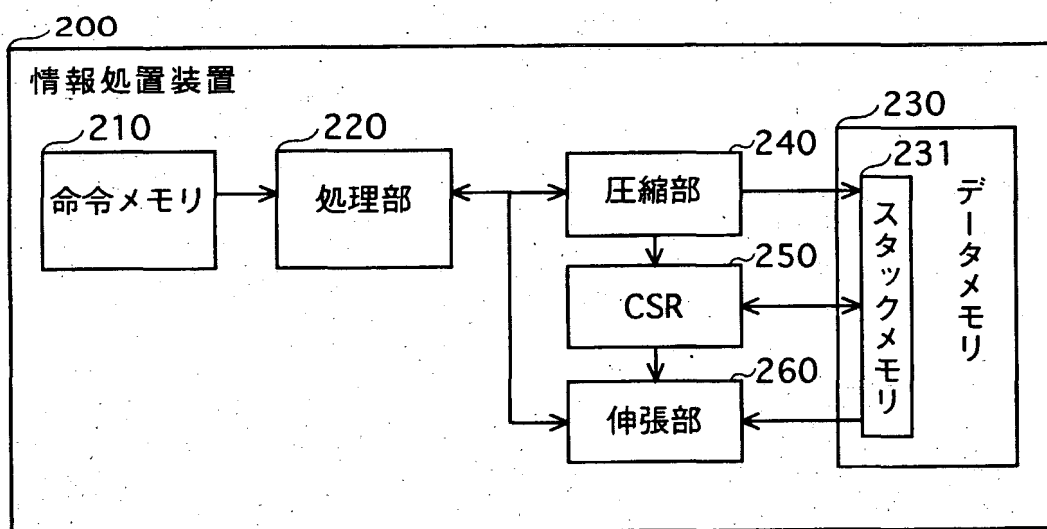
【図 8】



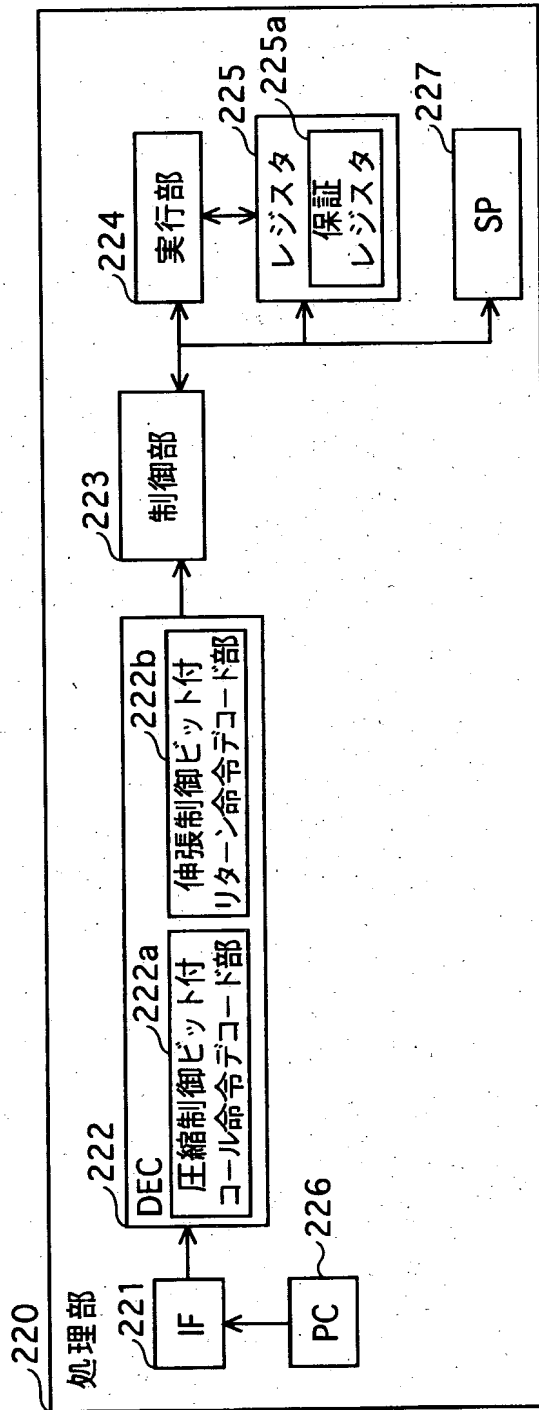
【図9】



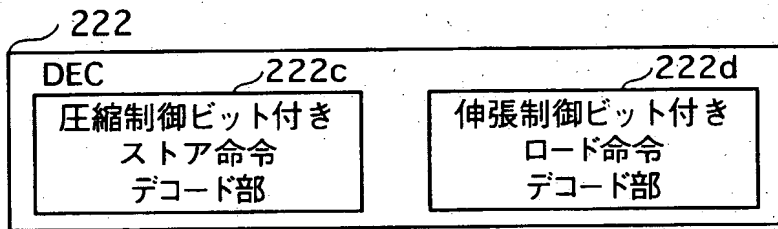
【図10】



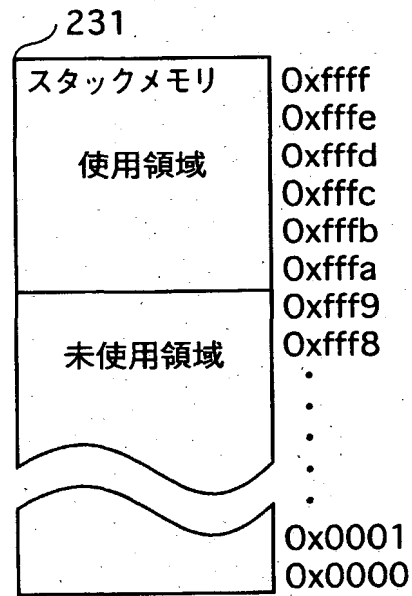
【図 11】



【図 1 2】



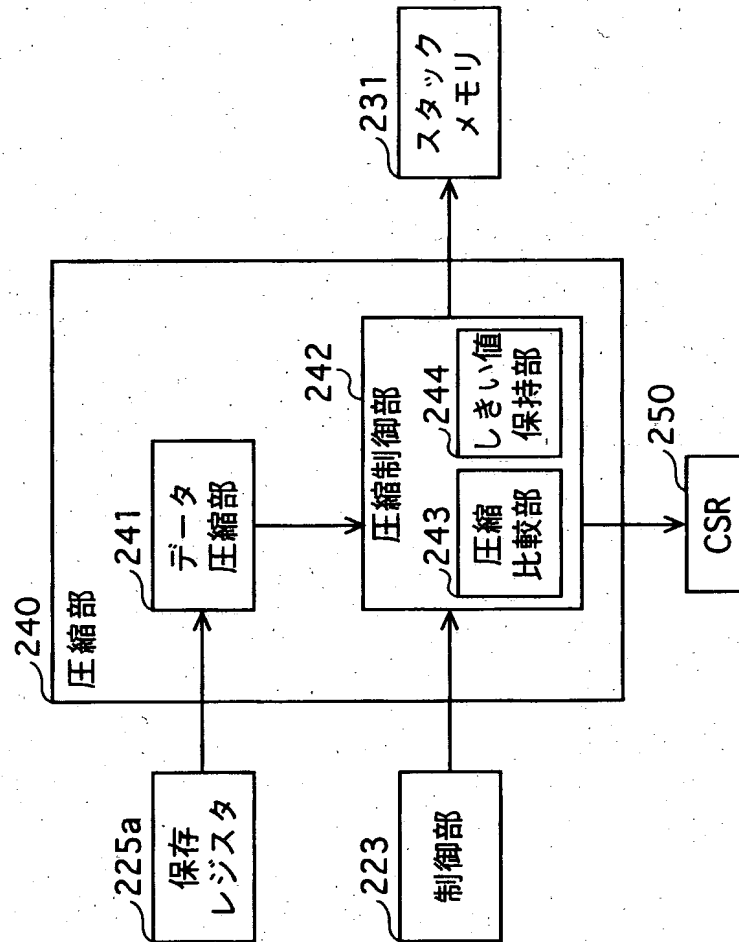
【図 1 3】



【図 1 4】

PC
r0:0x80000110
r1:0x00000000
r2:0x0000FFFF
r3:0x00000000
r4:0x80000010
r5:0x80000014
r6:0x50000010
r7:0x50000000

【図15】



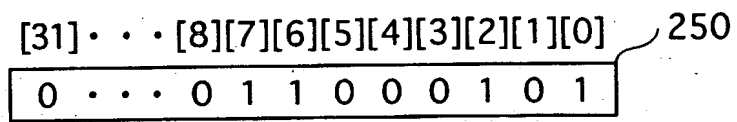
【図 1 6】

値	ハフマン符号
0x4	0b00000
0x5	0b00001
0x8	0b0001
0xf	0b001
0x1	0b01
0x0	0b1

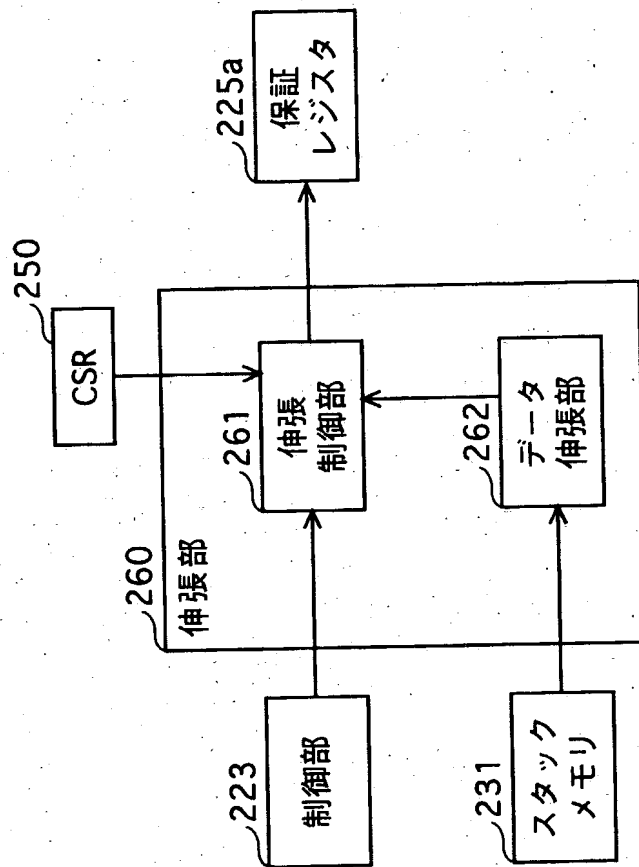
【図 1 7】

PC
0x1F5FFF92
0x4FF8F08F
0x0007EC3F
0111
CSR

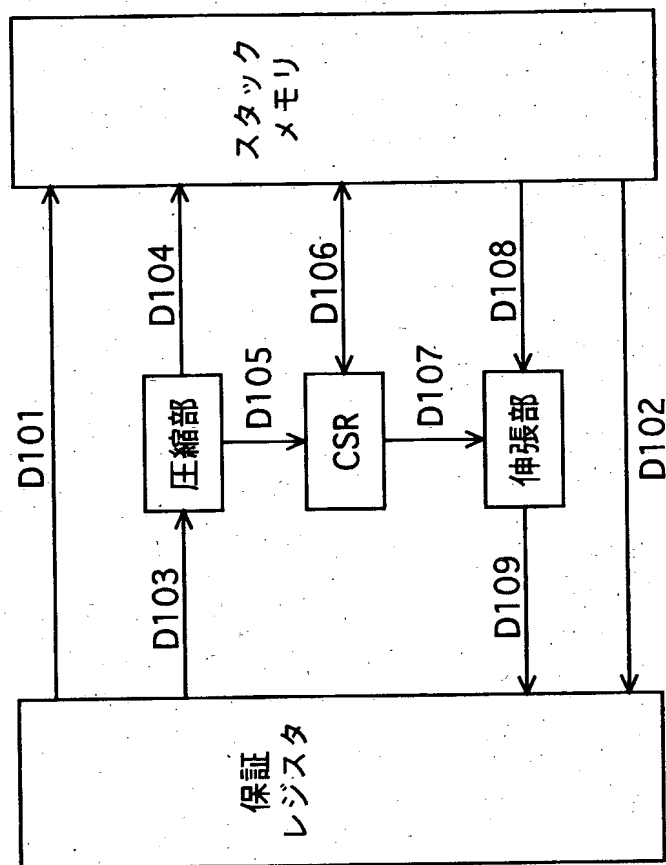
【図18】



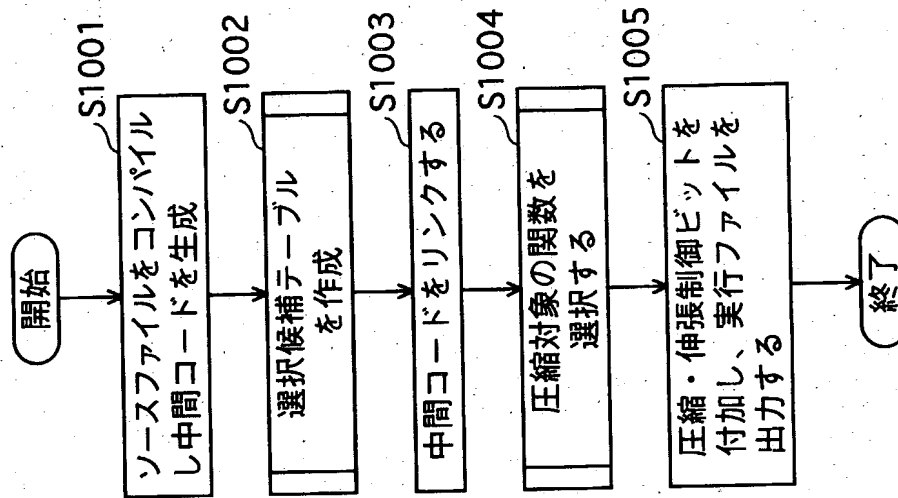
【図19】



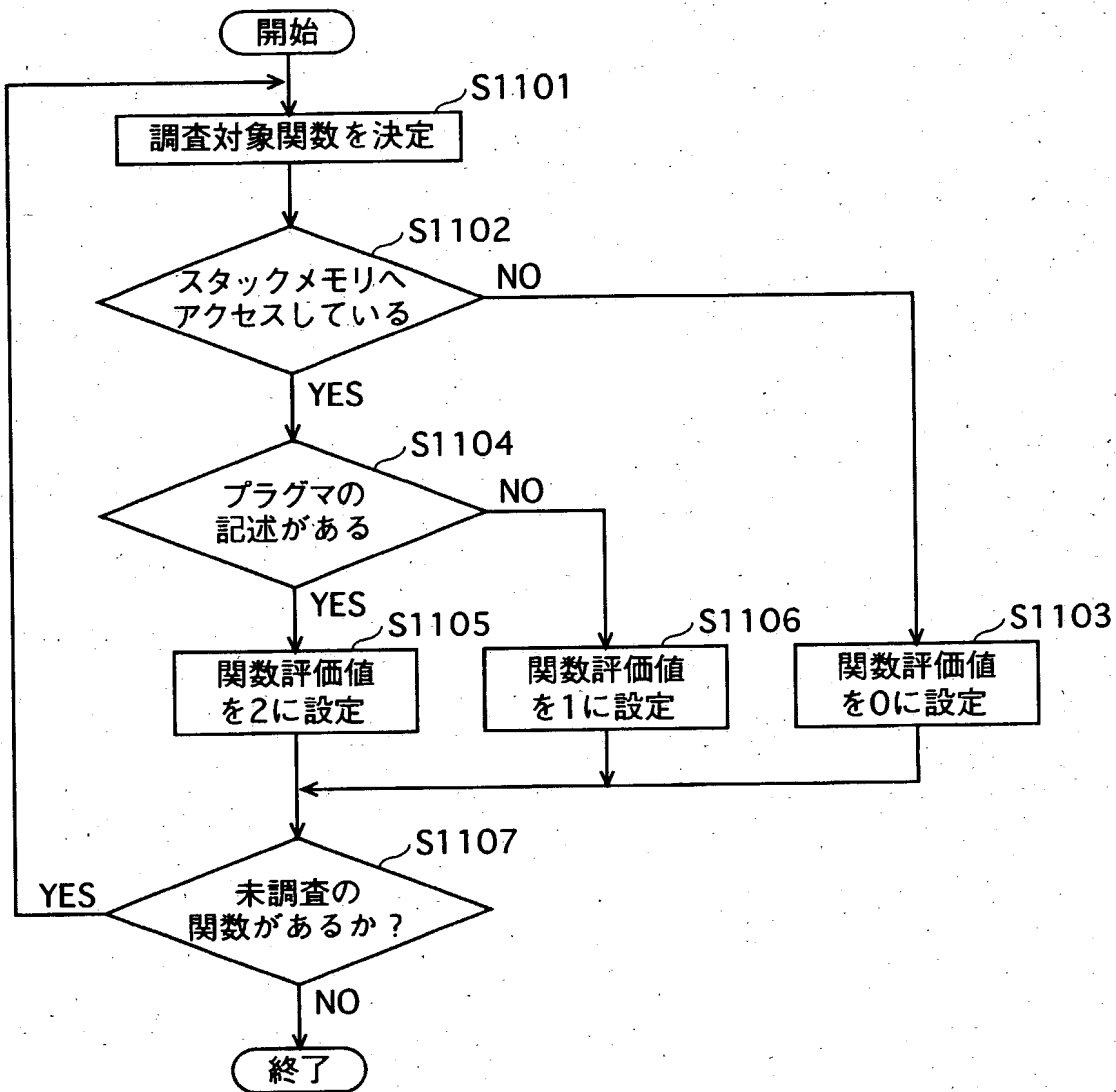
【図 20】



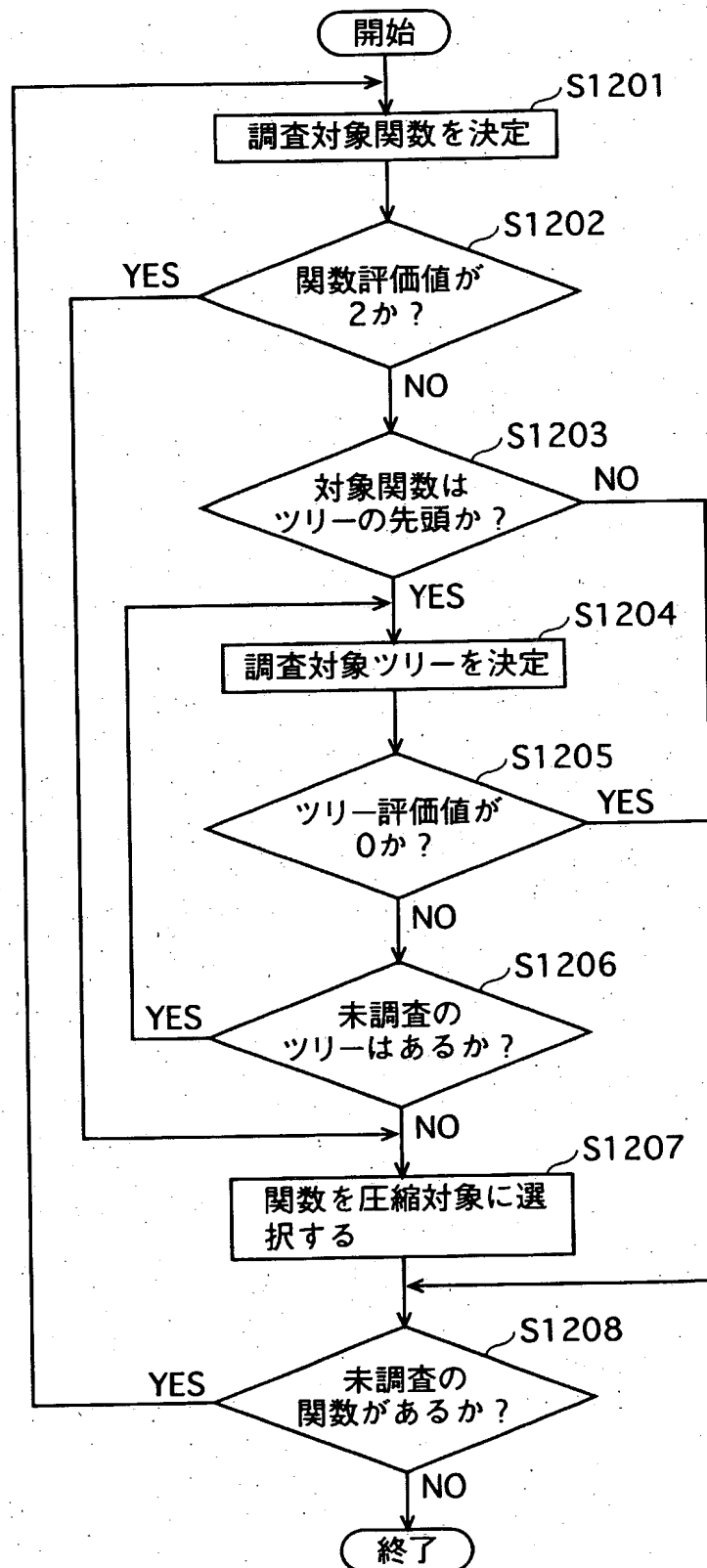
【図 21】



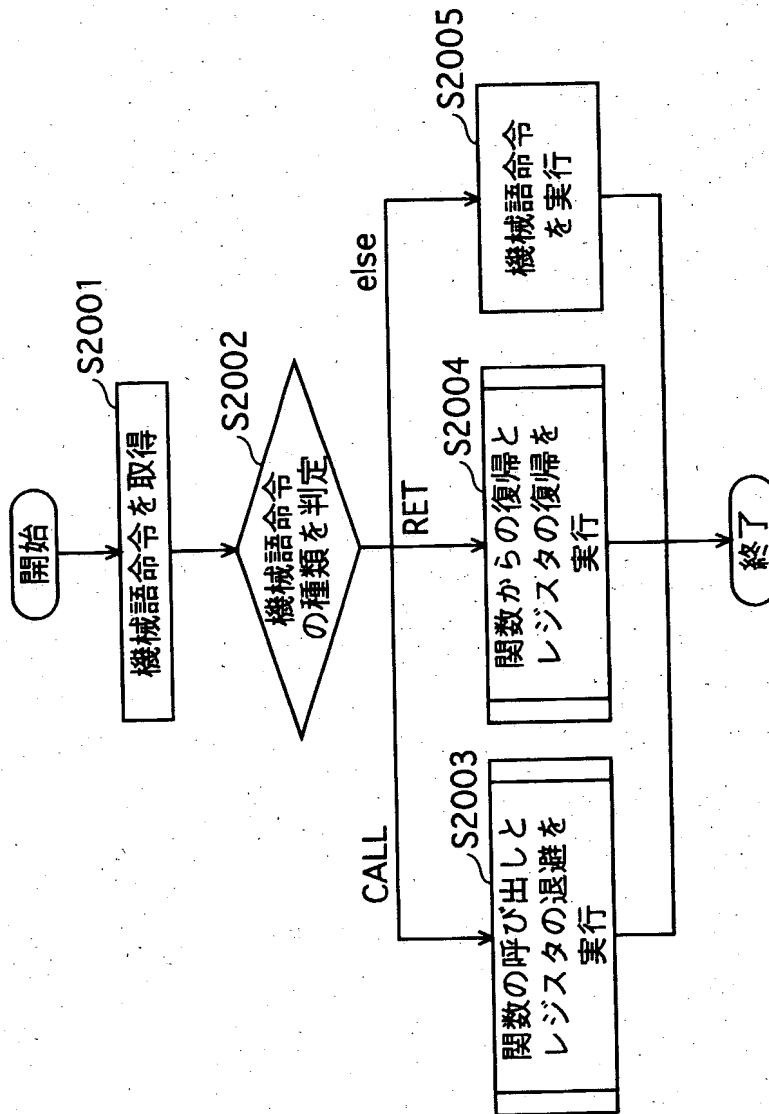
【図 22】



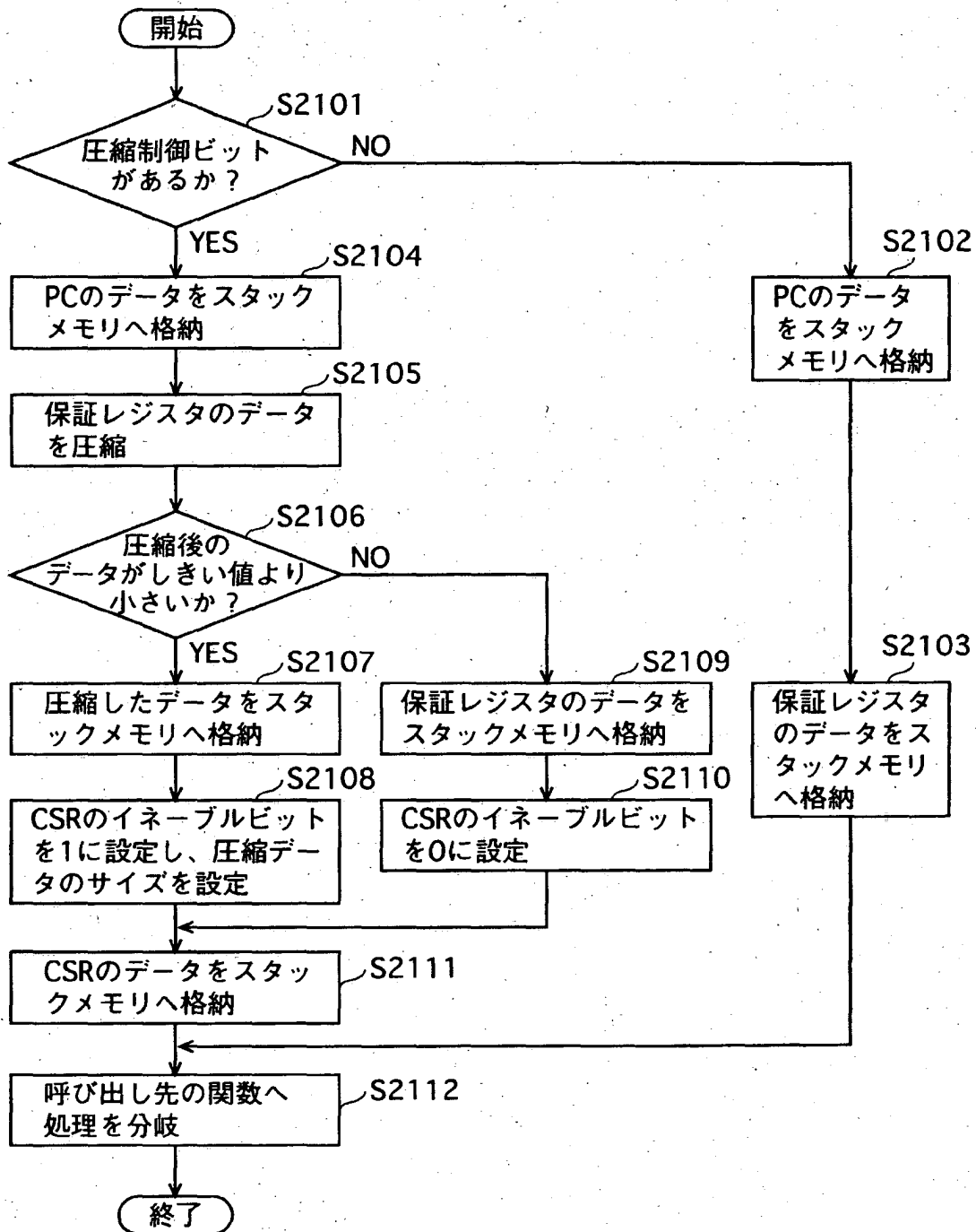
【図 23】



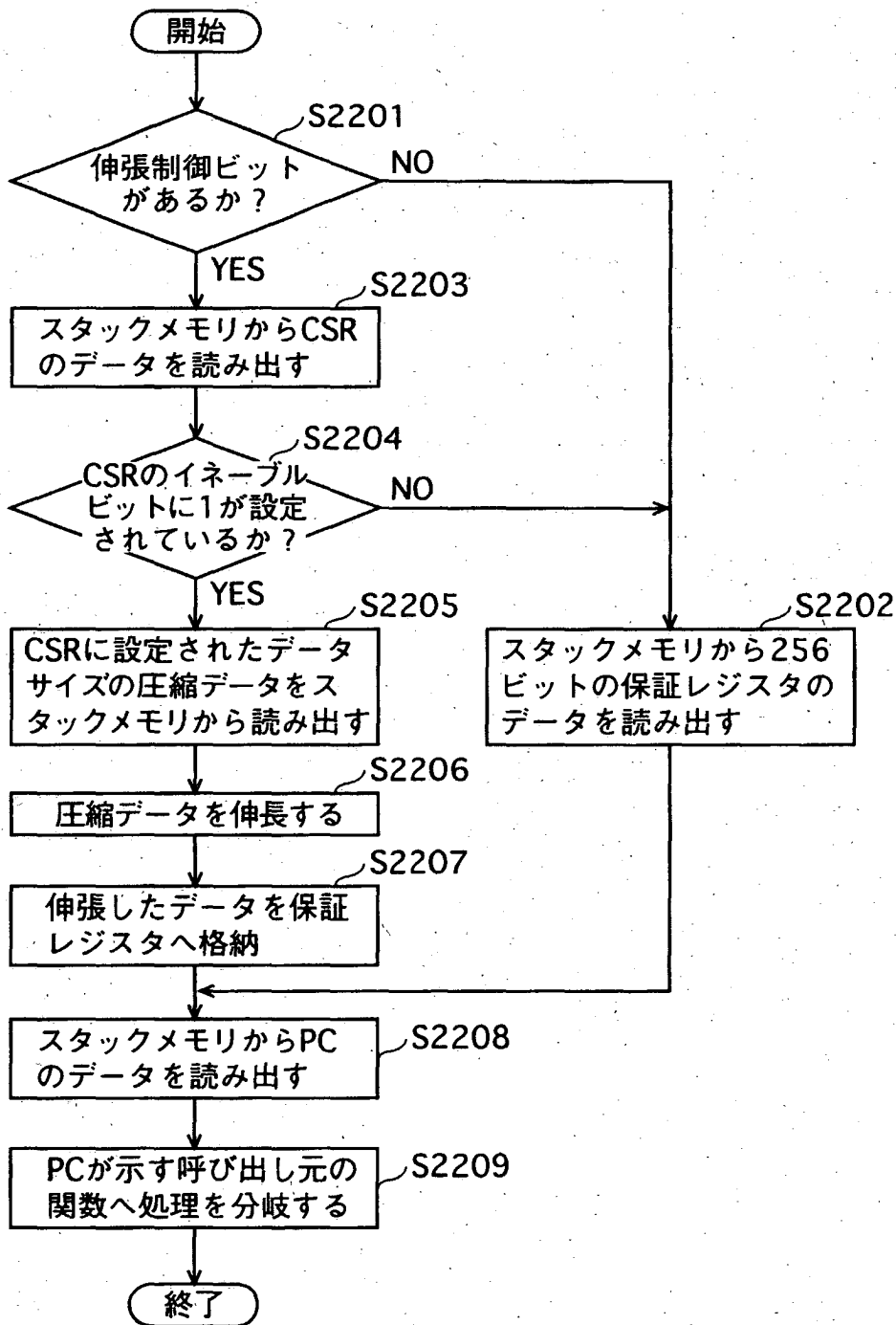
【図 24】



【図 25】



【図26】



【書類名】 要約書

【要約】

【課題】 情報処理装置においてスタックメモリの使用効率を高める。

【解決手段】 情報処理装置 2 0 0 であって、所定の関数の呼び出しに伴いスタックメモリ 2 3 1 へ保証レジスタの内容を圧縮して退避することを示す圧縮情報が機械語プログラム中に有るか否かをデコーダにおいて識別し、圧縮情報が有る場合に保証レジスタの内容を圧縮してスタックメモリ 2 3 1 へ退避する圧縮部 2 4 0 と、所定の関数の呼び出し終了に伴い保証レジスタへスタックメモリ 2 3 1 の内容を伸張して復帰することを示す伸張情報が機械語プログラム中に有るか否かをデコーダにおいて識別し、伸張情報が有る場合に、スタックメモリ 2 3 1 の内容を伸張して保証レジスタへ復帰する伸張部 2 6 0 とを備える。

【選択図】 図 1 0

出 願 人 履 歴 情 報

識別番号 [000005821]

1. 変更年月日 1990年 8月28日

[変更理由] 新規登録

住 所 大阪府門真市大字門真1006番地

氏 名 松下電器産業株式会社